Congestion Pricing as Scalable, Efficient and Stable Congestion Control for Future IP Networks

Dissertation

Sebastian Zimmermann

2005

This dissertation has been published as a book by VDE-Verlag (http://www.vde-verlag.de).

Acknowledgments

The results presented in this dissertation were obtained during my work at the Department of Communication Networks of the University of Technology at Hamburg-Harburg from 1999 to 2003. I would like to express sincere thanks and gratitude to my supervisor ("Doktorvater") and head of the department, Prof. Dr. Ulrich Killat, for his support and productive discussions as well as disputes. I also wish to thank Prof. Dr.-Ing. Ralf Lehnert, who was co-evaluator of this dissertation, and Prof. Dr. Hermann Rohling, who chaired the examination committee.

I am indebted to my former colleagues at the Department of Communication Networks, who were always extremely helpful and made the stay at the department a very pleasant experience. Special thanks go to Dr.-Ing. Kai Below for his support.

Various simulations and experiments presented in this dissertation were conducted by students as part of their project work. They also took part in the enhancement of the network simulators. I owe them many thanks for their assistance. In particular, I want to mention Tilo Hamann, who significantly contributed to the research presented in Chapter 5 as part of his Master's thesis.

Parts of this work were financially supported by the Deutsche Forschungsgemeinschaft (DFG).

Last but not least, I would like to thank my family and my fiancé for their encouragement and support.

Abstract

This dissertation focuses on the design of distributed congestion control algorithms for TCP/IP networks that are more powerful by utilizing the theory of Congestion Pricing as a mathematical framework. Currently implemented congestion control algorithms have several drawbacks that lead to sub-optimal usage and unfair distribution of network resources. Further, new applications have significantly changed the demand for network performance and quality of service. As will be shown, the use of wireless links and large link capacities can cause instability of the algorithms in use today.

Congestion Pricing is a strategy based on economics and optimization theory: A congestion measure (shadow price) is computed at each network node and fed back to the source. The sources adapt their rates according to utility functions and aggregate pricing information. It can be shown that this will lead to a social optimum for the entire network while maintaining both low queue sizes and high utilization. By choosing the user's utility function, different classes of service can be implemented without additional network support. Thus, the increasing demands on the network can be met without changing the Internet's fundamental principle of keeping the network nodes simple, and thus retaining its flexibility and scalability. At the same time, efficiency will also be significantly improved.

In this dissertation, Congestion Pricing will be applied to the Transmission Control Protocol (TCP). First, an implementation is developed that makes use of the full pricing information. Then, in an effort to make the new TCP source compatible with the existing network and TCP receivers, the pricing information is reduced to a single bit. This reduction of information introduces new challenges that are addressed by a "Single Bit Resource Marking (SBRM)" proposal developed by the author of this dissertation. Its performance is evaluated by comparison with other proposals and current congestion control algorithms.

While the efficiency and scalability problems are solved by the application of Congestion Pricing, a control theoretic model is further developed to examine the linear stability of the proposed algorithms. Current congestion control algorithms can become unstable in realistic scenarios, which significantly harms network performance. SBRM, in contrast, is stable over a wider range of network scenarios, and the impact on performance parameters is lower in cases of instability.

Lastly, multimedia applications are also addressed in this dissertation. They usually cannot change their transmission rate. Therefore, a distributed Call Admission Control using Congestion Pricing is developed. Even without special network support, it will be effective and highly efficient. Since the Call Admission Control works in a distributed manner to make it scalable, it can be implemented in network border gateways or in the sources themselves.

Contents

List of Acronyms

ACK	Acknowledgment
AQM	Active Queue Management
a-RED	Adaptive RED
ADSL	Asymmetric Digital Subscriber Line
ARPA	Advanced Research Projects Agency
ATM	Asynchronous Transfer Mode
CA	Congestion Avoidance
CAC	Call Admission Control
CP	Congestion Pricing
CP-TCP	Congestion Pricing based TCP
CWND	Congestion Window
CWR	Congestion Window Reduced
DARPA	Defense Advanced Research Projects Agency
DNS	Domain Name System
ECE	ECN Echo
ECN	Explicit Congestion Notification
ECT	ECN Capable Transport
EPF	Explicit Price Feedback
FAST	Fast AQM Stable TCP
FTP	File Transfer Protocol
HS-TCP	High-Speed TCP
HTTP	Hyper-Text Transfer Protocol
ICMP	Internet Control Message Protocol
IETF	Internet Engineering Task Force
IP	Internet Protocol
ISDN	Integrated Services Digital Network
FIFO	First In First Out
LAN	Local Area Network
MDDS	Maximum Datagram Data Size
MPEG	Motion Picture Expert Group
MPLS	Multi-protocol Label Switching
MSS	Maximum Segment Size
MTU	Maximum Transfer Unit
NCP	Network Control Protocol
RED	Random Early Discard/Detection

REM	Random Exponential Marking
RM	Resource Marking
RTO	Retransmission Timeout
RTP	Real-time Transport Protocol
RTT	Round-Trip Time
SACK	Selective Acknowledgment
SBRM	Single Bit Resource Marking
SDSL	Symmetric Digital Subscriber Line
SLA	Service Level Agreement
SS	Slow Start
SST	Slow Start Threshold
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VQM	Virtual Queue Mechanism
WAN	Wide Area Network
WTP	willingness to pay
XCP	eXplicit Congestion control Protocol

List of Variables[1]

variable	unit	description
N	1	Number of sources
L	1	Number of links
$x_n(t)$	$\frac{pkts}{s}$	Rate of source n
$y_l(t)$	$\frac{pkts}{s}$	Load at link l
$U_n(x_n)$	$rate$	Utility of source n at rate x_n
$B_n(x_n, p_n)$	$rate$	User's benefit
$C_l(y_l)$	$rate$	Cost function of link l with load y_l
$p_n(t)$	1	End-to-end path price seen at source n
λ_l	1	Shadow price at link l
w_n	$\frac{pkts}{s}$	Willingness to pay of source n
κ	$\frac{1}{s}$	Gain of TCP source
$\overline{\kappa}$	1	$\overline{\kappa} := \kappa \cdot RTT$
$cwnd_n(t)$	$pkts$	Congestion window of source n
$RTT_n(t)$	s	Round-trip time observed at source n
c_l	$\frac{pkts}{s}$	Capacity of link l
b_l	$pkts$	Buffer occupancy at link l
$m_l(t)$	1	Marking probability at link l
γ_l	$\frac{1}{pkts}$	Scaling factor at link l

[1]Where possible, same variable names and units are used throughout this dissertation. However, because of the wide variety of algorithms described, changes in notation could not completely be avoided. They will be noted in each section where they occur.

Chapter 1

Introduction

The Internet, successor of the ARPANET, is a world wide computer network driven by a protocol suite commonly known as “TCP/IP”. This protocol suite was officially deployed on the Internet in 1983 [Pos81b], creating a standard for the interconnection of different computer and local area network architectures. The Internet Protocol (IP) was already developed in 1978 to allow the addressing of computer systems at any network connected to the former ARPANET. Still earlier, in 1974, the Transmission Control Protocol (TCP) was introduced. It replaced the Network Control Protocol (NCP), which was the first transport layer protocol of the ARPANET.

TCP has four main purposes: The first purpose is the establishment and tear-down of logical connections between two hosts as well as the segmentation of data transmitted between them. In addition, TCP ensures reliability of the transmissions by transparently retransmitting segments that were not received correctly. The third purpose is flow control. It is required to prevent overflow of the receiver’s buffers. Flow control is different from congestion control, which is the fourth purpose of TCP. Since the Internet lacks a call admission control, the network may become overloaded. TCP adapts its sending rate automatically according to the network state.

The TCP/IP protocol suite has become extremely successful. However, when these protocols were developed, the Internet was mostly a research network connecting a few hundred computers of various universities. Data link capacities were rather small. Nowadays, the Internet is an international network connecting more than 170 million hosts [ISC03] in households, companies and universities. The capacity of the data transfer links has grown dramatically and will continue to grow in the future. New kinds of hosts are connected to the Internet such as mobile phones and ubiquitous devices. TCP/IP was not designed for such growth. For example, while twenty years ago four bytes seemed more than sufficient for addressing the hosts, IP addresses are now considered a scarce resource. Links were considered to have very small bit error rates, which is not the case with wireless links that are widely used today. Growing link capacities and router queue memory sizes also have a negative impact on TCP’s fairly complex rate control loop. And finally, the applications on the Internet are changing. Until a few years ago, the world wide web, e-mail, network news, and file transfer were the only dominant applications. All of these applications are server/client based and use TCP. Today, peer-to-peer and interactive multimedia applications are becoming popular. They pose different demands on the network. For example, interactive multimedia traffic requires low latencies, and TCP’s automatic retransmissions are thus counterproductive. The alternative protocol used today for multimedia traffic, the User Datagram Protocol (UDP), completely lacks built-in congestion control. Since the proportion of UDP traffic on the Internet is growing, congestion control algorithms for this type of traffic have to be developed.

In this dissertation, drawbacks of current congestion control algorithms and previously as well as newly proposed modifications will be examined. It will further be demonstrated why current TCP cannot be used in modern high-bandwidth networks. Congestion control is an active field of research, and many changes to TCP and new links algorithms in the router queues have been proposed to address some of the problems of the current TCP. While most additions to TCP have been proposed as singular “fixes” to some of the problems of TCP, in this dissertation another approach will be applied that makes use of a mathematical framework derived from economics and optimization theory. It can be used to holistically redesign TCP and the link algorithms for optimal performance.

This alternative approach, called “Congestion Pricing”, is based on cost and utility functions. The cost functions describe link costs when they are busy or even overloaded. The utility functions describe the benefit of the user when he can use the network at a certain rate. Using these functions, the network can be optimized for maximum utility and minimum cost, for example minimum queuing delay. While this optimization problem would be trivial if it could be solved centrally, in practice this is not possible since the current network state and users’ preferences are not known. Even if one built such a central controller, it would not scale and therefore not be able to handle the large number of flows on the Internet backbones. However, it was shown that the optimization problem can be modified to a “distributed optimization problem”, where each host on the network solves a separate optimization problem. The user information is available at the end systems, thus the only problem is the transport of network state and cost information to the hosts.

This mathematical framework can be applied to solve a large number of problems with TCP, eventually leading to the desired “Scalable, Efficient and Stable Congestion Control” for IP networks and the Internet specifically. In this dissertation it will be shown how the mathematical framework can be applied to TCP. Practical implementation issues and compatibility with current TCP will also be addressed as well as the resulting performance improvements. Congestion Pricing can also be used for other protocols such as UDP to implement a rate control, and it can be used for *distributed* call admission control. This will be examined in the last part of this dissertation.

To further motivate the topic of this dissertation and to present the background, in Chapter 2 current TCP variants and the problems of their congestion control algorithms will be presented. Some changes and additions will also be discussed that have been proposed to solve some of the problems. They will be used in the remainder of this dissertation as base for performance comparisons.

In Chapter 3, the Congestion Pricing framework will be introduced. The mathematical optimization problem will be presented, and how a distributed version also solves the global optimization problem. This framework is the mathematical basis for the algorithms that will be used throughout this dissertation.

To apply Congestion Pricing theory to TCP, significant changes are necessary. A TCP variant that makes full use of Congestion Pricing theory will be developed in Chapter 4. It will then be compared to current TCP variants to demonstrate its superior performance. This version, however, is not compatible with current IP networks such as the Internet. Only limited information can be used because current protocols do not allow additional fields. Therefore in Chapter

5, ways to apply Congestion Pricing to the Internet will be presented. Different approaches will be compared, finally leading to an entirely new approach that outperforms all other variants and conventional TCP by far. This new TCP variant, “Single Bit Resource Marking (SBRM)”, can be used to solve almost all of the problems with current TCP.

Only recently, control theoretic analysis of Internet protocols became an active field of research. Such models can be used to evaluate linear stability of congestion control algorithms. A control theoretic model that was developed for SBRM will be introduced in Chapter 6. Using the model, scalability of SBRM with regard to stability can be evaluated. Although SBRM may also encounter stability problems, they occur later and with less impact than what is observed with conventional TCP. The model can be used to better understand SBRM and to develop even better algorithms.

In Chapter 7, Congestion Pricing theory will be applied to multimedia traffic. Two new approaches for congestion control with these traffic types will be proposed: A distributed Call Admission Control (CAC) and dynamic quality adjustment of streaming media as a reaction to congestion signals. It will further be shown that Congestion Pricing can be well applied to multimedia traffic. However, additional modifications are required.

Finally, to conclude, the presented methods and results will be summarized in Chapter 8. Furthermore, some potential applications and further improvements that have not been addressed within this dissertation will be presented.

Chapter 2

Congestion Control Background

2.1 Congestion Control in Packet Switched Networks

In packet switched networks, when multiplexing different data streams, temporary overload conditions can be resolved by queuing packets until the router or the output line becomes available again (cf. Figure 2.1.1). However, the average load has to stay well below 100%. If the

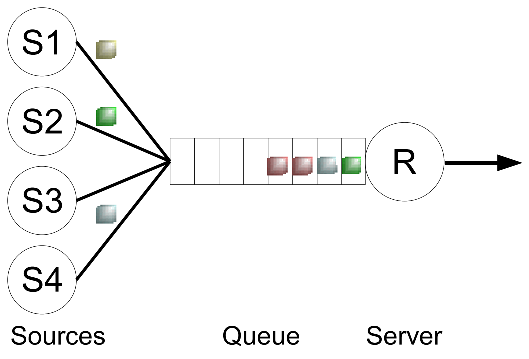


Figure 2.1.1: Multiplexing four sources

load is higher, the queues will continue to grow and overflow. But also temporary overload conditions can be bad for some types of applications, for example real-time applications, as the queuing introduces an additional delay. Some networks control the load by implementing a *Call Admission Control (CAC)* and by policing flows according to previously announced traffic parameters (for example the Integrated Services approach by the IETF [BCS94]). In such a case, transmission rates of every stream are fixed or can only vary within a range described by traffic parameters. Bandwidth for every flow is reserved in advance via a *Resource Reservation Protocol (RSVP)*. Thus, this is an *open-loop* control approach. This type of approach has a significant disadvantage. Every router has to keep track of all traffic parameters of every flow passing through it. It also has to police every flow to ensure that the traffic parameters are not violated. This requires the storage of large amounts of state information, and therefore generates scalability problems. Alternative concepts such as the *Differentiated Services* approach [LR98] solve the scalability problem by focusing on traffic aggregates and by providing only *Quality of Service (QoS)* classes, but cannot prevent overflows as single flows are not policed, and again generally only an *open-loop* control approach is used to decide whether a new flow is accepted.

So-called *best effort* networks do not keep any state information at all. There is no guarantee that a packet is delivered. There are no upper bounds for delays, and re-routing is possible. These networks always accept new traffic, but in case of overload, packets are discarded. Generally, best effort networks perform well while the load is clearly below the network's capacity. Without any congestion control, every user would try to use the full bandwidth he needs. But since overload of the network is very likely, such behavior on the part of all users would at some stage lead to a *congestion collapse* (cf. Figure 2.1.2). In case of a congestion collapse,

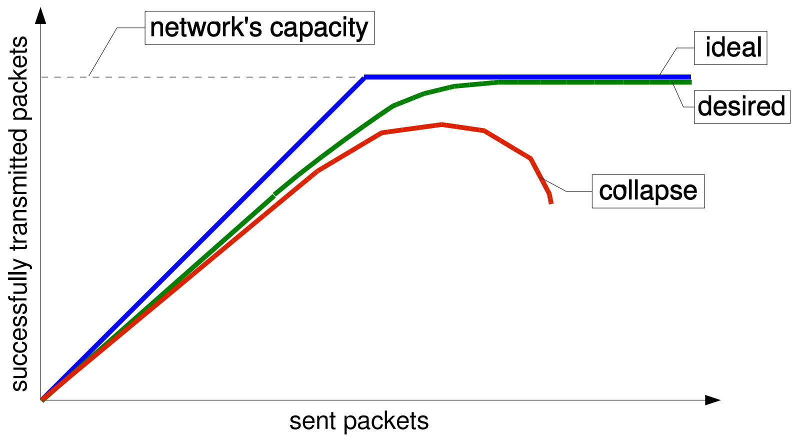


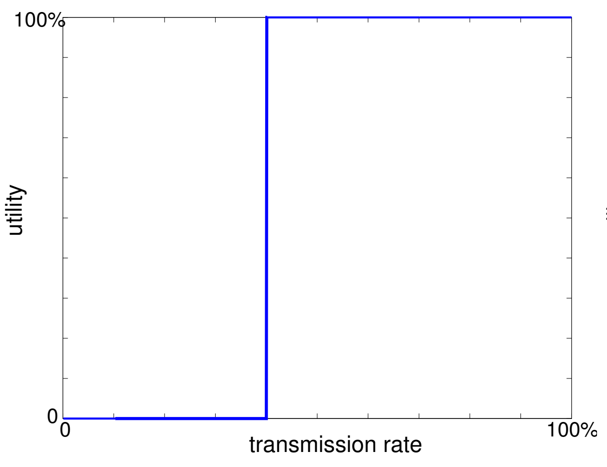
Figure 2.1.2: Congestion collapse

the effective throughput becomes worse, although the offered load is increased. For this reason, congestion control is necessary to ensure reduction in load before a congestion collapse can occur. For best effort networks, a *closed-loop* control approach is used. Whenever a source notices packet loss due to overload, it will reduce load on the network by reducing its transmission rate. Thus, packet losses are negative feedback signals in the congestion loop. The advantage of this type of congestion control is its scalability. It is solely implemented in the sender and receiver. No network interaction is needed — besides the packet drops in case of overflowing queues. Additionally, because of the *closed-loop* control, reactions to changing network conditions are possible. This approach to congestion control has been implemented in the *Transmission Control Protocol (TCP)* [Pos81c], the dominant transmission layer protocol on the Internet and other *Internet Protocol (IP)* [Pos81a] based networks.

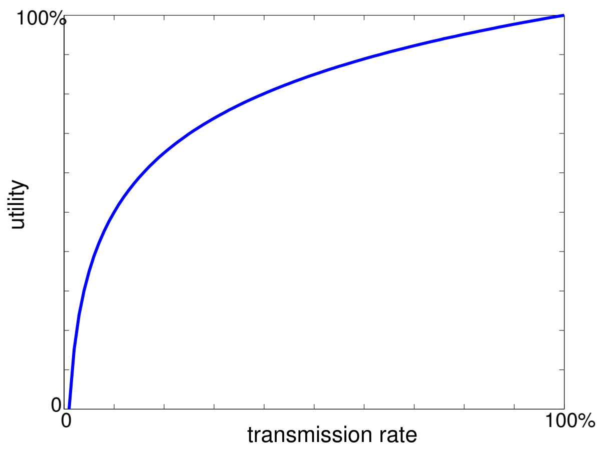
2.2 Elastic Traffic vs. Inelastic Traffic

Throughout the dissertation, two types of traffic will be distinguished: *elastic traffic* and *inelastic traffic* [She95]. Inelastic traffic cannot change its transmission rate. For example, voice traffic is encoded at a certain data rate and thus requires a matching transmission rate. For this kind of traffic, a *Call Admission Control (CAC)* is usually employed. In telephony networks, the call admission control prevents overloading of the networks and ensures reliable service, but sometimes a telephone call is blocked and the caller hears a special busy signal. On the Internet, however, call admission controls do not exist. Until now, this was not a problem, as the predominant traffic type is elastic traffic. Elastic traffic can adjust its rate and therefore respond to congestion. E-mail is an example for an application that generates elastic traffic. If the network is overloaded, the e-mail will be transferred at a lower speed. It does not really matter whether it takes two seconds or one minute for the e-mail to arrive, however, a faster arrival is better than a slower arrival.

Both types of traffic can be described by utility functions. They show the relationship between the rate at which traffic can be transmitted and the user's corresponding utility. Obviously, for inelastic traffic the user's utility is zero when the available rate is below the required rate. The utility is 100% when the available rate rate matches the required rate, and it will not increase when the available rate exceeds the required rate. This is modeled by a step function as shown in Figure 2.2.1a. For elastic traffic, however, the utility will increase as the available rate increases. Usually it is assumed that for elastic traffic the utility function is concave. Under this condition, the user's utility increases at slower pace than the transmission rate. This is shown in Figure 2.2.1b.



(a) Inelastic traffic



(b) Elastic traffic

Figure 2.2.1: Utility functions for inelastic and elastic traffic

Since the *Transmission Control Protocol (TCP)* [Pos81c] adjusts its transmission rate automatically, it is a good transport layer protocol for elastic traffic, but bad for inelastic traffic. Thus, TCP is usually used by applications and corresponding protocols such as web browsers (HTTP [FGM$^+$99]), e-mail (SMTP [Pos82]) , news readers (NNTP [KL86]), and file downloads (FTP [PR85]). TCP is still the dominant protocol on the Internet, accounting for more than 95% of the bytes transmitted [Sch03]. Even according to the estimated number of flows, TCP dominates with 65 – 70%. Consequently, to optimize the use of network resources, it is important to optimize TCP's congestion control algorithms.

The remaining 5% of bytes transmitted are due to the *User Datagram Protocol (UDP)* [Pos80]. UDP is a simple protocol that neither adjusts the transmission rate nor ensures reliability of the transmissions. It is predominantly used for short transfers such as name resolution (DNS [Moc87]) and time updates (NTP [Mil92]). However, since it does not control the transmission rate, UDP is also used for all types of inelastic traffic including multimedia streams. An increase of UDP's bandwidth share can therefore be expected in the future, as multimedia applications are becoming more popular in IP networks.

2.3 Congestion Control Mechanisms in Conventional TCP Variants

2.3.1 Properties of TCP

As mentioned in the previous Section 2.2, TCP is the dominant transport layer protocol on the Internet. Usually, TCP is used by an application through a byte-stream socket. The payload is automatically segmented such that it can be stored in an IP packet. The size of each payload segment is at most the Maximum Segment Size (MSS), which depends on the Maximum Datagram Data Size (MDDS) or the respective Maximum Transfer Unit (MTU), that the network can support [Pos83]:

$$\begin{aligned} MTU &= MDDS + sizeof(\text{TCP-header}) \\ &= MSS + sizeof(\text{TCP-header}) + sizeof(\text{IP-header}). \end{aligned}$$

Additionally, in contrast to UDP, TCP opens a virtual connection between both hosts and ensures reliable and ordered data transfer by sorting received segments and retransmitting lost segments (cf. Figure 2.3.1).

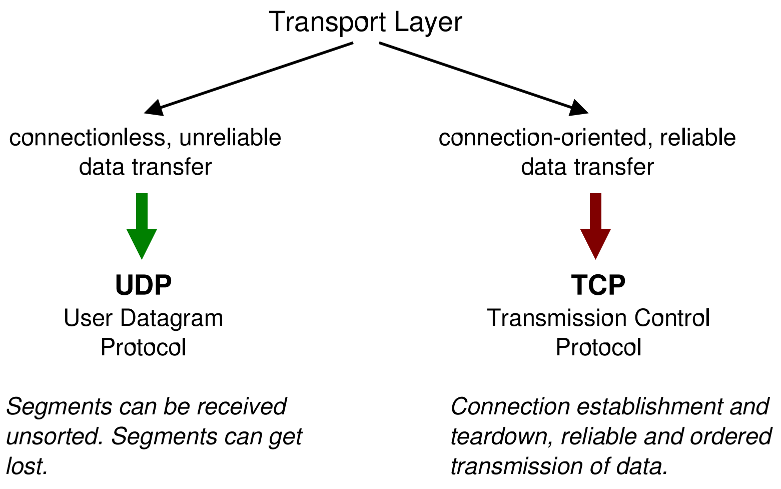

Transport Layer
connectionless, unreliable data transfer
connection-oriented, reliable data transfer
UDP
User Datagram Protocol
TCP
Transmission Control Protocol
Segments can be received unsorted. Segments can get lost.
Connection establishment and teardown, reliable and ordered transmission of data.

Figure 2.3.1: UDP vs. TCP

Figure 2.3.2 shows the structure of a TCP packet. To ensure reliability, all segments are numbered and the reception is acknowledged. The sequence number is the number of the last byte of the payload. Whenever the receiver correctly receives a TCP packet, an acknowledgment is generated that contains the first byte of missing (or expected) payload. For the purpose of this dissertation, the unit bytes for segment and acknowledgment numbers is not used. Instead, just use ordinal numbers are used. If all packets are using the maximum payload size, both values are related by the maximum segment size (MSS). Additionally, in contrast to the standard, the acknowledgment acknowledges the last segment that was received in order and correctly, not the next expected one. This is done to simplify explanation without changing functionality, and a common practice found in textbooks on networks.

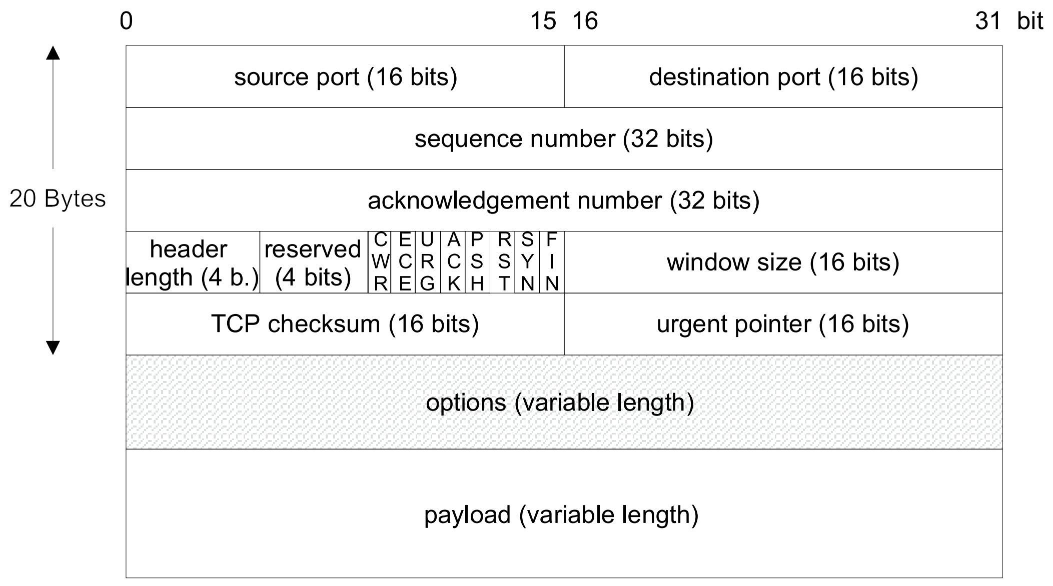


Figure 2.3.2: Structure of a TCP packet

TCP is a sliding window protocol. The sliding window, the so-called *transmission window*, limits which segments may be transmitted. Only segments that lay within the transmission window may be transmitted. When a segment at the lower end of the window is acknowledged, the window will slide such that a new segment enters the window and may be transmitted. The size of the transmission window is determined by the *congestion window (CWND)*. Again, according to the standard, window sizes are given in bytes. Here the number of segments is used instead. Congestion window sizes can be translated from number of segments to number of bytes by using

$$cwnd_{bytes} = MSS \cdot cwnd_{segments}.$$

The maximum window size without special options is 64 kilobytes. For simplicity, a MSS of 1 kByte is used; thus in the presented examples the maximum window size is 64 segments.

Since the matching acknowledgment for a transmitted segment is received roughly one round-trip time later, the window size is approximately equivalent to the number of segments that can be transmitted during one round-trip time. Thus the sending rate can be estimated by:

$$x(t) \approx \frac{cwnd(t)}{RTT(t)}. \tag{2.3.1}$$

This relation will be used throughout the dissertation.

2.3.2 TCP's Fundamental Algorithms

In this subsection, the conventional algorithms that change the size of the congestion window [APS99] and hereby adapt the transmission rate will be presented.

Slow Start (SS)

When a new connection is established, the size of the congestion window is set to one segment, and the *slow start threshold (SST)* is set to 32 segments. As long as the size of the congestion window is below the slow start threshold, the *slow start* algorithm is used. For every segment that is correctly acknowledged, the congestion window size will be increased by one segment. Thus, after one round, when a full window of segments has been transmitted and acknowledged, the congestion window size, and thus also the depending transmission window size, would have doubled (cf. Figure 2.3.3). Although the exponential growth of the congestion window size

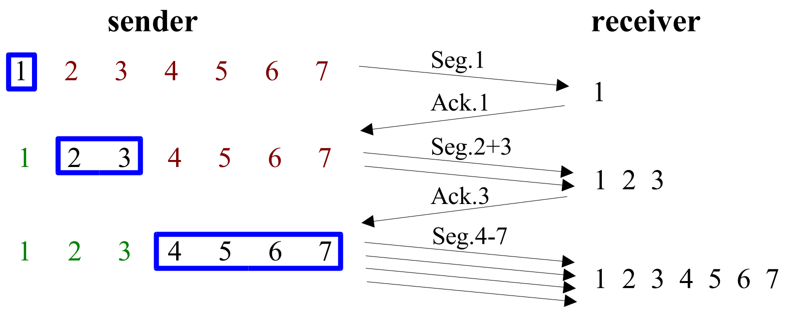


Figure 2.3.3: Growth and advancement of the transmission window

should perhaps be called "fast", the slow start algorithm is named "slow start" because the congestion window size is initially one. If the receiver uses so-called *delayed ACKs* [Cla82, Bra89], generally only every second segment is acknowledged, thus leading to a growth of the congestion window that is only half as fast as without delayed acknowledgments.[1]

When a segment is lost, it will not be acknowledged. The absence of the acknowledgment will then be detected by a timeout, the so called *Retransmission Timeout (RTO)*. After such a timeout, the slow start threshold is set to half of the current congestion window size, but never less than two segments, and slow start is entered again.

Congestion Avoidance (CA)

As soon as the congestion window size reaches the slow start threshold, another algorithm, the *congestion avoidance* algorithm, is entered. In this case, the congestion window is only increased by one segment when a full window of segments has been acknowledged. Thus, the growth is now linear as shown in Figure 2.3.4. When packet loss is detected by a retransmission timeout, the slow start threshold is adjusted to half of the current congestion window size, and slow start is entered.

Fast Retransmit

With *fast retransmit*, the sender utilizes the fact that the receiver acknowledges all segments that it receives. For each segment that arrives after a lost one, the receiver will again acknowledge

[1]Some TCP implementations will detect that the acknowledgment covers two segments and increase the congestion window just like two acknowledgments would have done. This is an option that was added later to fight manipulated TCP stacks that would try to increase the sending rate by generating additional acknowledgments [All03].

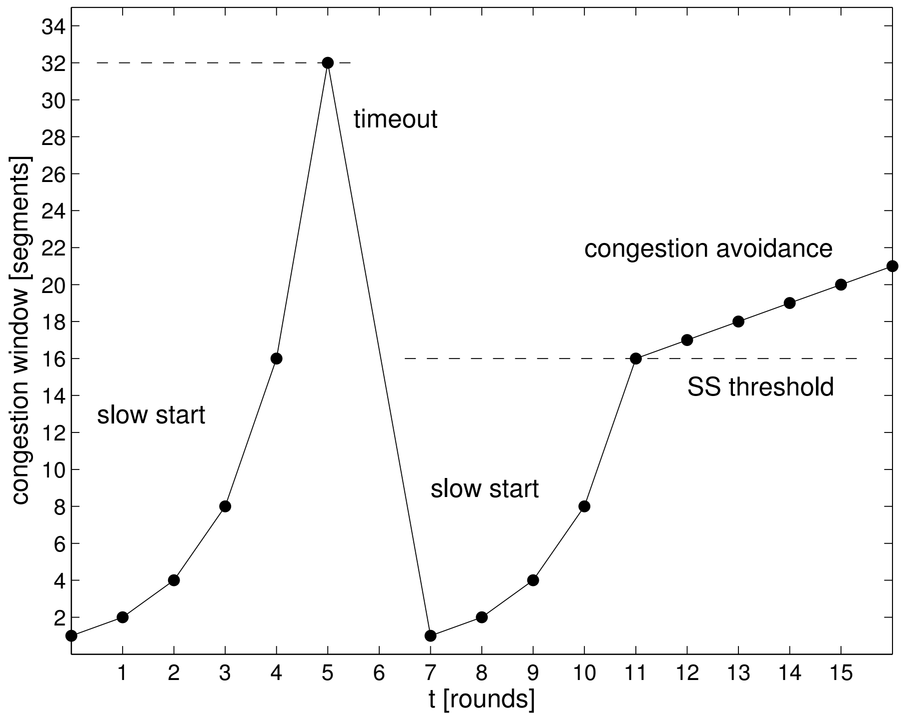


Figure 2.3.4: Schematic progression of the congestion window

the last segment that was received before the lost one. For example, the sender sends segments 1, 2, and 3. Segment 1 is received and triggers an acknowledgment for itself. Segment 2 is lost, but segment 3 is received. Segment 3 cannot be acknowledged, since segment 2 is still missing. Instead, segment 1 is acknowledged again. Because of this, the sender can detect the missing packet by the duplicate acknowledgments. Usually, a retransmission is triggered when the third duplicate acknowledgment arrives. This is called *fast retransmit*. Since the previously used retransmission timeout (RTO) is significantly larger than a round-trip time (RTT), fast retransmit will detect packet losses much faster than the timeout mechanism. However, since three duplicate acknowledgments are required for fast retransmit to work, the congestion window must be at least the size of four segments (cf. Subsection 2.4.1). After a fast retransmit, the slow start threshold is set to half of the current congestion window, and slow start is entered.

Fast Recovery

The *fast recovery* algorithm modifies the behavior after a fast retransmit. Previously, after a retransmission, the congestion window was reduced to one segment and slow start was entered. With fast recovery, however, the congestion window is halved and congestion avoidance is entered after reception of an acknowledgment of the retransmitted segment. Additionally, while the retransmitted segment is still unacknowledged, the sender stays in the fast recovery phase. During this phase, the congestion window is also increased for every duplicate acknowledgment because an acknowledgment indicates that some packet was received and thus left the pipe. This is called *inflation* of the congestion window, and will speed up recovery. It is important to note that fast recovery is only entered after a fast retransmit. Thus, packet loss must be detected by duplicate acknowledgments. Retransmission timeouts still significantly degrade throughput.

TCP SACK option

The *Selective ACKnowledgment* (SACK) option is another solution to the multiple packet loss problem. Using an optional TCP header, the receiver can tell the sender exactly which acknowledgments were received after a missing one. It was proposed in 1996 [MMFR96].

2.3.3 Important Variants of TCP

After describing the fundamental algorithms of TCP, the conventional TCP variants will be briefly introduced:

TCP Tahoe

TCP Tahoe appeared in 1988 in the 4.3 BSD Tahoe release. This was the first release that used the fast retransmit algorithm in addition to slow start and congestion avoidance. There are also TCP Tahoe versions that do not implement the fast retransmit algorithm. In the remainder of this dissertation, these versions will be referred to as "old" Tahoe. Most Windows operating systems use "old" Tahoe.

TCP Reno

TCP Reno adds the fast recovery algorithm to TCP. It was implemented in the 4.3 BSD Reno release of 1990. Most UNIX operating systems use this variant. TCP Reno is the first TCP variant that fully implements an *"additive increase multiplicative decrease (AIMD)"* strategy of the congestion window. When an acknowledgment is received, the congestion window is increased *linearly* by adding $\frac{1}{cwnd}$ (congestion avoidance algorithm). But when the loss of a packet is detected, the congestion window is halved, thus the decrease is *multiplicative*. This behavior is depicted in Figure 2.3.5. Note that the increase is only roughly linear. Since the round-trip time (RTT) increases when the queue size increases, the congestion window update rate does not grow as fast as the congestion window, leading to a concave curve.

TCP NewReno

TCP NewReno, first proposed in 1995/1996, is identical to TCP Reno, however, it uses an improved fast retransmit algorithm. The fast recovery algorithm allows TCP to recover more quickly from multiple packet losses. With normal TCP Reno, the fast recovery phase ends with the first acknowledgment that acknowledges new data, and the congestion window is deflated. In case of multiple packet losses, this acknowledgment will only acknowledge the segment before the second segment that was lost. This is called a *partial acknowledgment*. The modified fast recovery algorithm will only partially deflate the congestion window and continue the fast recovery phase, if the received acknowledgment is a partial acknowledgment [FH99]. This version is used in most modern operating systems.

TCP Vegas

While the classical TCP variants Tahoe, Reno, and NewReno detect congestion by packet loss, TCP Vegas, proposed in 1994/1995, uses an entirely different approach. TCP Vegas measures

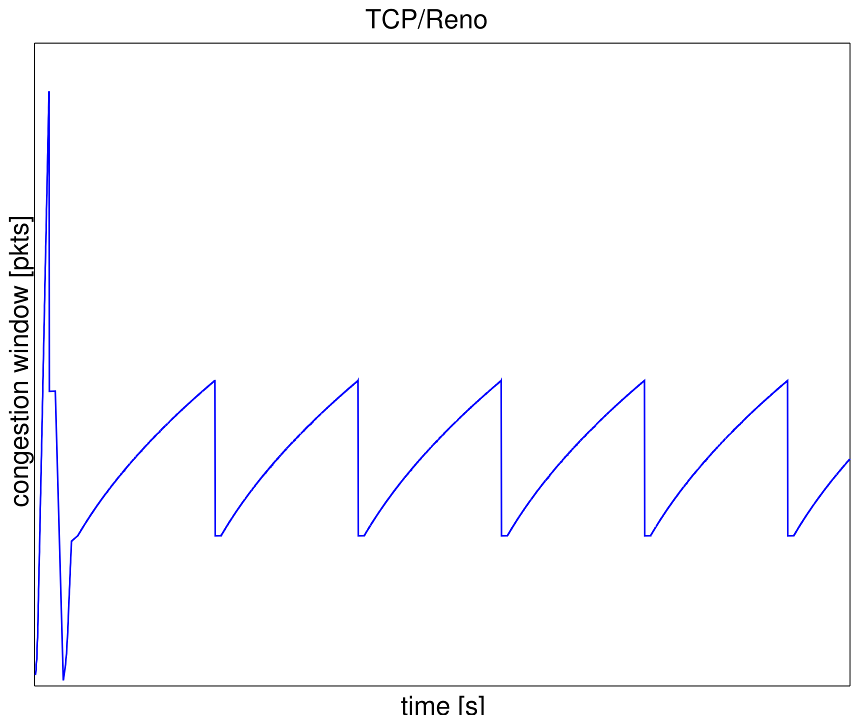


Figure 2.3.5: AIMD property of TCP Reno

the current round-trip time (RTT) at a high resolution and also keeps the encountered minimum RTT. Full queues will increase the round-trip time, thus this can be taken as measurement of congestion.

Similar to (2.3.1), the current transmission rate is calculated as follows:

$$x_{actual} = \frac{cwnd}{RTT_{current}}.$$

At the same time, the expected transmission rate is calculated:

$$x_{expected} = \frac{cwnd}{RTT_{min}}.$$

If the difference $x_{expected} - x_{actual}$ is greater than a threshold β, the congestion window is reduced. It is increased if the difference is smaller than a threshold α [BP95]. While this approach is very appealing as it reduces transmission rate when delay increases and before packet loss occurs, there are significant problems related to round-trip time changes caused by possible rerouting and load balancers, as well as fairness issues [MLAW99]. TCP Vegas has never been widely employed, but can optionally be activated in Linux since the kernel version 2.6.6 (2004).

2.4 Drawbacks of TCP and Proposed Extensions

2.4.1 Major Drawbacks of Conventional TCP Implementations

The transmission rate reduced only after a segment has been lost due to overload.

Except for TCP Vegas, all other conventional TCP versions will reduce the transmission rate only after a segment has been lost. Thus, overload had already occurred. A good TCP variant should be able to avoid overload conditions.

Lost segments have to be retransmitted, thus causing a reduced goodput.

Packet losses are used to signal overload. The source will reduce the sending rate, but has to retransmit the lost segments. In case of overload, packet losses are more likely, and more packets must be retransmitted, adding additional load to the bottleneck link. Thus, from a certain threshold, goodput will drop again as overall load increases.

The detection of lost segments is slow or requires a minimum congestion window size.

Generally, lost segments are detected by a retransmission timeout (RTO): If the corresponding acknowledgment does not arrive before the retransmission timer expires, the segment is considered lost. Such a timeout is usually very slow ($\gg RTT$) to allow arrival of the acknowledgment even in cases of delay. Thus, reaction to congestion is slow, too. Alternatively, a lost segment is detected by three duplicate acknowledgments acknowledging the last correctly received segment. For this to be possible, the congestion window size must be at least four:

1. Lost segment; no acknowledgment is generated
2. Correctly transmitted segment, generating first duplicate acknowledgment of the previously received segment
3. Correctly transmitted segment, generating second duplicate acknowledgment
4. Correctly transmitted segment, generating third duplicate acknowledgment

Most TCP versions do not recover well from multiple packet losses.

Conventional router queues for the Internet are drop-tail queues. They will drop every packet that arrives when the maximum queue capacity is exceeded. Since TCP tends to send in bursts, it is relatively likely that more than one packet is dropped during a period of congestion. Without SACK option, however, TCP can only detect the first packet that was lost. Subsequently lost segments can only be detected after the first retransmitted segment has been acknowledged, which adds additional delay to the detection and recovery process. Also, there must still be enough packets in transit that three duplicate acknowledgments can be triggered. Otherwise, retransmissions will only occur after a slow retransmission timeout.

TCP cannot distinguish between losses due to congestion and due to transient errors.

Packet loss is always interpreted as a congestion signal. However, transient packet losses due to transmission errors are also possible. For conventional wire based networks, this is not a problem as congestion losses are much more likely than transient losses. With modern networks this poses a problem. The TCP transmission rate in bytes per second, BW, as a function of the packet loss probability p and the maximum packet size in bytes B is given by [FF99]:

$$BW \leq \frac{1.5\sqrt{\frac{2}{3}}B}{RTT\sqrt{p}}. \tag{2.4.1}$$

Thus, for networks that allow high transmission rates, the packet loss probability must be very low. In high speed networks it therefore is possible that the transient loss probability is in the same order of magnitude as the congestion loss probability. In such a case, the sources will unnecessarily reduce the transmission rate instead of just retransmitting the segment that was lost due to the transmission error. Even worse, in wireless networks transient losses are much more likely. Thus, conventional TCP is neither suitable for networks with very high capacities nor for wireless networks.

The rate allocation depends on the round-trip time.

The TCP bandwidth formula (2.4.1) also reveals another property: the bandwidth allocation is inversely proportional to the round-trip time (RTT). Thus, if two connections compete for the same bottleneck link and one connection has a larger round-trip delay, it will receive a smaller share than the other connection. This property does not only introduce unfairness, but can also lead to starvation of connections with large round-trip times if load is high while the other connections have significantly smaller round-trip times.

Deterministic drops may lead to the global synchronization problem.

Conventional drop-tail queues drop packets deterministically when the capacity is exceeded. For this reason, it is likely that several packets from different connections will be dropped at the same time. All these connections will halve their transmission rate at the same time, causing the queue to empty again. If the round-trip times are also roughly equivalent, all these connections will then increase their transmission rate at the same speed until the overload condition is reached again. This type of synchronization leads to severe low frequency oscillations (cf. Figure 2.4.1). Using control theoretic models, this intuitive explanation of global synchronization can also be shown mathematically. In Chapter 6, such a control theoretic model will be presented.

Direct coupling of the packet loss probability or queue size and the congestion measure is problematic.

With conventional TCP, the packet loss probability is used as congestion measure. Further, the reaction to a congestion signal is fixed. The transmission rate of a single source is always halved. If many sources use one bottleneck link queue, the overall reaction to packet loss will be less than if only a few sources use the bottleneck link queue (cf. Figure 2.4.2). Thus, the

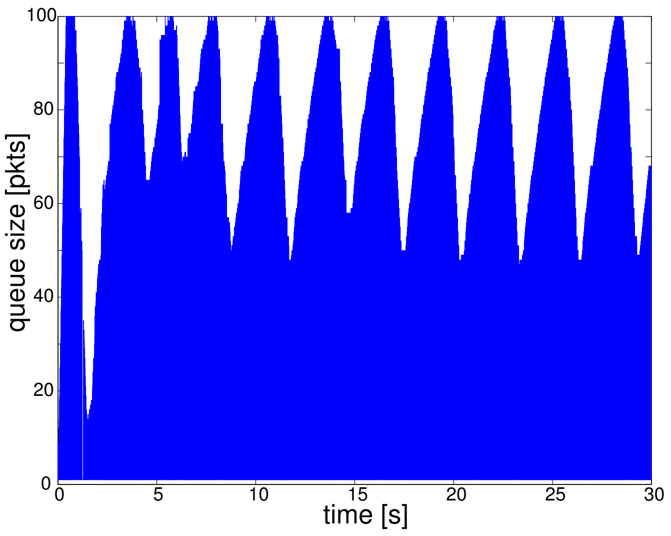
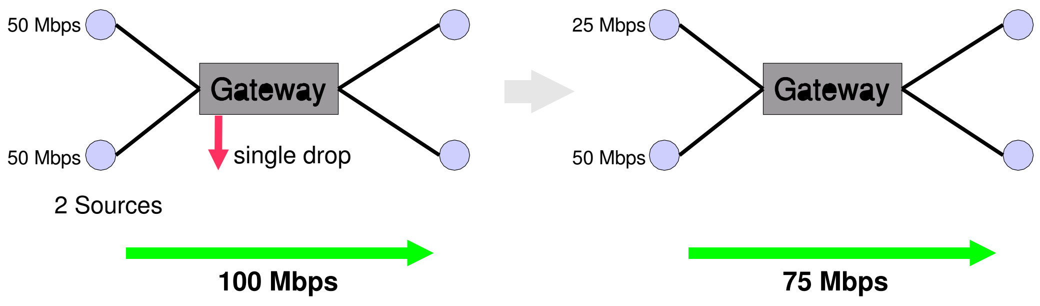
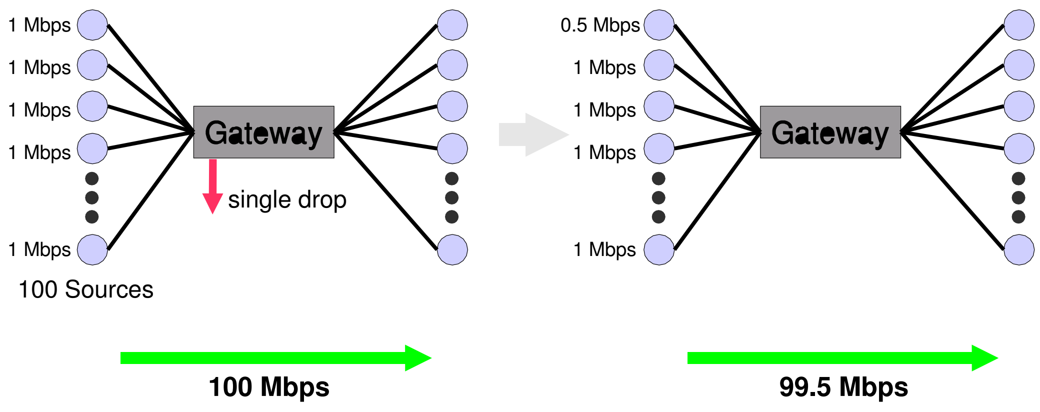


Figure 2.4.1: Oscillations caused by global synchronization



(a) Reaction to a single drop when two sources are active



(b) Reaction to a single drop when 100 sources are active

Figure 2.4.2: Change in resulting rate when reaction to packet loss is the same regardless of the number of sources (as in TCP Reno)

congestion measure must be "stronger" if many sources share the same link. A "stronger" congestion measure in this case is a higher packet loss probability, leading to bad utilization. Even with so-called *Active Queue Management (AQM)*, where packets are dropped randomly before the queue becomes empty, the dropping probability usually depends on the queue size. This leads to increased persistent queue size if many sources share the link.

TCP and drop-tail queues will lead to full queues and high variance.

Since all conventional TCP versions, except for TCP Vegas, only reduce the transmission rate after packet loss has occurred and otherwise further increase the transmission rate, it is normal that conventional drop-tail queues will exceed capacity. Thus, drop-tail queues in front of a bottleneck link will usually maintain a persistent queue size and therefore a persistent queuing delay. This behavior is not desired. The sole purpose of queues in the core network is to provide a possibility to temporarily store bursts of packets until the outgoing link becomes free again.

Further, since TCP is designed to reduce the transmission rate in a conservative manner by halving, it is very likely that the queue size will become small before the next congestion cycle begins. A high variance of the instantaneous queue size is therefore likely (also cf. Figure 2.4.1). As consequences, bottleneck link utilization will degrade if the queue runs empty; and queuing delay varies, introducing *jitter* which is bad for real-time multimedia applications.

The optimal queue capacity is difficult to tune.

Since conventional TCP variants tend to fill queues, large queue capacities will also lead to large average queue sizes if the link is a bottleneck link. If the queue capacity is too small, packet loss is likely. Lost packets have to be retransmitted, leading to decreased goodput. Further, multiple packet losses will cause timeouts, leading to under-utilization of the bottleneck link. If the queue runs empty, utilization will also degrade. Thus, a network operator has to choose between good bottleneck link utilization and low queuing delays. Achieving both is not possible. Commonly, network providers choose queues with large capacities in front of bottleneck links to increase utilization. As a trade-off for improved utilization and throughput, persistent delay is increased.

TCP does not allow service differentiation or Quality of Service (QoS).

The current TCP/IP based networks are *best-effort* networks: all traffic is treated the same, and there is no guarantee that a packet will be received. However, since different applications have different demands on the network, this may not be sufficient. New applications require *Quality of Service (QoS)* guarantees or at least service differentiation.

2.4.2 Explicit Congestion Notification

Explicit Congestion Notification (ECN) [RFB01] was proposed in order to solve the problems related to packet loss as a congestion measure. Packet loss as a congestion signal is replaced by codepoints in the IP header: When a router detects congestion, and the source supports Explicit Congestion Notification (ECN), which is indicated by the *ECN-Capable Transport (ECT)* codepoint in the IP header, the router will set the *Congestion Experienced (CE)* codepoint, but not drop the packet. The receiver of a packet that has the CE codepoint set will then send a TCP acknowledgment with the *ECN Echo (ECE)* flag being set and continue to set the ECE flag in all subsequent acknowledgments. When the original sender receives such an acknowledgment, it will reduce the sending rate just like if the packet was lost, but not retransmit that segment. It will also set the *Congestion Window Reduced (CWR)* flag in the next outgoing segment to indicate that it has received the ECE flag and has reacted. When the receiver receives a segment with the CWR flag set, it will discontinue to set the ECE flag in the acknowledgments. Such a handshake procedure increases robustness against loss of acknowledgments, but only allows for one congestion event per round-trip time.

The ECT and CE codepoints consist of two bits in the IP header which formally used to be part of the *Type-of-Service (TOS)* field. The use of two bits allows the signaling of four states:

1. Non-ECN-Capable flow
2. ECN-Capable flow
3. ECN-Capable flow, Congestion Experienced (1)
4. ECN-Capable flow, Congestion Experienced (2)

Although there are several proposals that use the second two states to signal different severities of congestion, the original proposal argues that full TCP compatibility is required to accommodate the incremental deployment of ECN [RFB01]. Since many core network devices are not yet ECN-aware and modify the TOS field in an incompatible manner, two equal *Congestion Experienced* codepoints are necessary to detect such incompatible modifications.

Nevertheless, other researchers argue that a congestion signal that causes a lesser reduction of transmission rate will be an incentive to deploy the new algorithms [Wel02]. M. Kwon and S. Fahmy [KF02] proposed an ECN(α/β) algorithm that only requires changes to the TCP source algorithm and modifies TCP's increase/decrease behavior to change the congestion window less aggressively when ECN congestion signals are received. Focusing on fairness, T. Hamann and J. Walrand [HW00] changed ECN to improve fairness with regard to round-trip time (RTT) dependency.

It becomes obvious that ECN alone cannot solve the major problems of conventional TCP. Notably, it adds a new means of transporting congestion information to the source that can be used for future congestion control algorithms. Such an approach will be introduced in Chapter 5. The original Explicit Congestion Notification is presented and analyzed more in detail in [Büc01].

2.4.3 Active Queue Management

Active Queue Management (AQM) was proposed to tackle the problem where transmission rate is only reduced after congestion, i.e. overflow of the queue has occurred. Further, as shown before, deterministic packet drops or congestion signals often lead to the global synchronization problem. AQM is also based on a *first-in-first-out (FIFO)* queue, but unlike drop-tail queues, packets are dropped or congestion signals are generated in a probabilistic manner before the queue's capacity is exceeded. Depending on the implementation, the packet drop probability depends on the instantaneous or the average queue size, and a certain threshold. Alternatively to dropping, packets can be marked if the transmission protocol supports congestion indication by marks (cf. *Explicit Congestion Notification*). Only the most commonly known AQM variant, which is already implemented in modern routers, will be introduced here:

Random Early Detection (RED)

The *Random Early Detection (RED)* AQM algorithm, originally called *Random Early Discard*, changes the packet dropping or congestion signal generating probability depending on the average queue size of the FIFO queue [FJ93]. When the average queue size is below a minimum threshold, all packets will enter the queue. When the average queue size grows beyond the minimum threshold, new packets entering the queue are dropped or marked with a certain probability $p_a(p_b)$. The probability p_b increases linearly with the average queue size if the average queue size is between a lower and an upper threshold, and becomes 1 if the upper threshold is exceeded (cf. Figure 2.4.3). The marking or dropping probability p_a is then calculated as

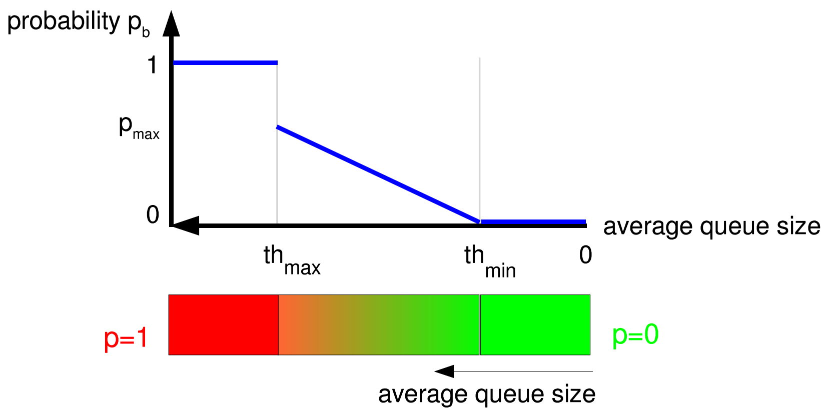


Figure 2.4.3: RED gateway: General principle

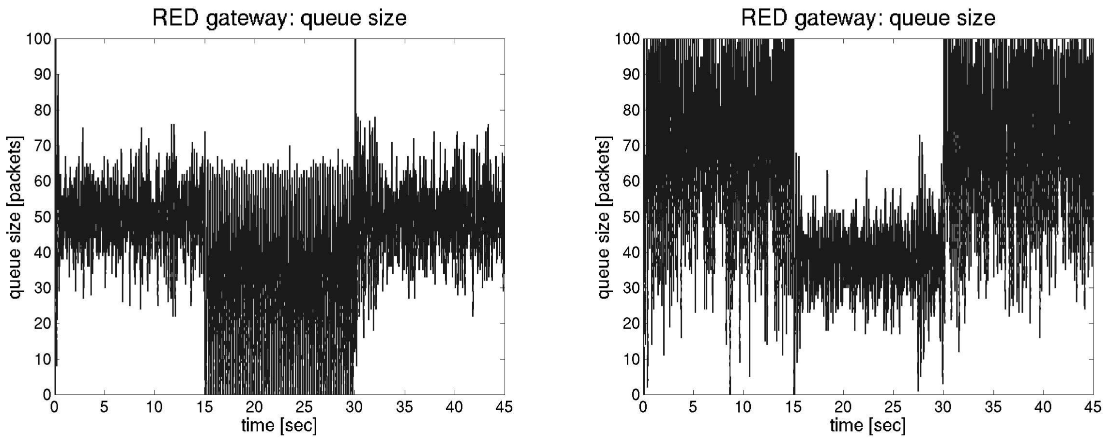
follows:

$$p_a = \min\left(\left[\frac{p_b}{1 - count \cdot p_b}\right]^+, 1\right),$$

where *count* is the number of packets that were received after the last packet that was dropped or marked, and $[x]^+ = \max(0, x)$. This is done to make the marking/dropping probability a uniform random variable. Note that RED requires four parameters to be tuned: minimum threshold th_{min}, maximum threshold th_{max}, maximum probability p_{max}, and weight of the moving average q_w.

Although a carefully tuned RED queue can reduce average queuing delay while keeping utilization high, and can avoid the global synchronization problem, it does not adapt well to changing network conditions. Figure 2.4.4 shows the development of the queue size over time. During the first 15 seconds, many sources are active and compete for the bottleneck link's capacity. Then, between the 15th and 30th second, some sources are turned off, thus reducing competition for the bottleneck link. After 30 seconds, again, many sources compete for the link.

Since every source is greedy and tries to send as much as possible, the bottleneck link should always be fully saturated, independent from the number of competing sources. Optimally the sum of all transmission rates should not exceed the bottlenecks capacity, thus leading to a low



(a) RED settings too aggressive (b) RED settings too conservative

Figure 2.4.4: Tuning of RED parameters

average queue size at all times. Figure 2.4.4a shows a RED queue where the RED parameters were chosen aggressively, such that marking probability is high and queue overflows are avoided. When the number of active sources is reduced between the 15th and 30th second, the queue runs empty, leading to under-utilization. If the RED parameters are changed such that under-utilization is avoided when only a few sources are active, the queue will overflow during periods where many sources are active (Figure 2.4.4b). Thus, while RED can be tuned for a certain network condition, it does not scale well with regard to the number of active sources.

Many other drawbacks of RED have been identified in the meantime, and several variants have been proposed. Some of them are given in [OLW99, FKSS99, MBDL99, FGS01, HMTG01a]. RED's characteristics are not in the scope of this dissertation, however, since RED is currently considered "best practice", it will be used to benchmark the performance of the proposed algorithms.

2.4.4 Time-stamp Option

To save resources on the end systems, TCP uses only a single counter for the measurement of the round-trip time (RTT) and the expiry of the retransmission timeout (RTO) timer. This single counter is incremented with a resolution of 500 ms, thus allowing only very rough estimates. To optimize retransmission timeout behavior, a *time-stamp option* [JBB92] was introduced that allows recording of the send time. It is then echoed by the receiver when sending the acknowledgment. This option, however, is not commonly implemented. Instead, modern TCP stacks use the old single counter with a resolution of 100 ms. The advantage of a high resolution timer is an optimized retransmission timeout. With a resolution of 500 ms, a retransmission timeout would at least last one second, leading to severe degradation of throughput when multiple packet losses occur.

This option will be used, as Congestion Pricing relies on good measurements of the round-trip time.

2.5 Conclusions

In this chapter the most important congestion control algorithms and TCP variants were presented. Several drawbacks exist, of which some have been addressed by extensions to TCP and the introduction of Active Queue Management. However, all of these improvements are only singular fixes to TCP. As a whole, TCP is not modified. Each proposal was developed and evaluated as single component because of the lack of a mathematical model that describes the full control loop including the sources, multiplexing, queuing, and congestion signaling. Evaluation for a network is possible using simulations, which can only be conducted in simple topologies as a result of computational limitations. To address the issues holistically, a mathematical framework is needed. *Congestion Pricing* is such a framework. In the remainder of this dissertation, the Congestion Pricing framework, modifications to TCP that use Congestion Pricing, and a control theoretic model describing the full control loop will be presented.

Chapter 3

Congestion Pricing Framework

In the previous chapter some important features and problems of current TCP implementations were presented. While several additions have been proposed each of which solve a single problem, their interdependence is often not well understood. For better insight and design of congestion control loops, it is advantageous to describe all parts, source algorithms, signaling and link algorithms, in a single model. Such a holistic model can be derived from the Congestion Pricing framework that will be introduced in this chapter.

3.1 Introduction and Motivation

A network should be designed in such a way that source and link algorithms work together to steer the network to a desirable operating point. Such a desired operating point refers to optimal utilization at low or even zero average buffer occupancy in the router queues. Thus, congestion control can be viewed as an optimization problem. Such optimization should be applicable in practice. While a central optimization method that maximizes overall utilization and minimizes queuing delay could easily be thought of, it could not be used in practice because it relies on instantaneous information of all users' demand for bandwidth, current use of bandwidth and buffer lengths. Even if one could signal this information to the centralized optimizer, scalability would still be a problem. Thus, another approach to the optimization problem is needed. The key idea in *Congestion Pricing* theory is that the resource sharing problem can be viewed as a distributed game. In this game, each user is a "selfish" participant that tries to maximize his own profit. If the game is designed correctly, this selfish behavior will lead to the optimal solution for all participants of the game and thus to the optimal solution of the resource sharing problem.

Problems like this are not only known from game theory, but also from economics. For example, pollution of the environment often does not impose a cost on the polluter, but on the general public. A financially selfish person therefore will not hesitate to cause pollution if it increases his profits. Similarly, on a network, overload will impose costs on every user, not just on the person who utilizes the capacity intensively. A selfish user will thus try to maximize his or her own transmission rate at the cost of all users. The key idea is now to charge each user for his share of the congestion: Each user does some "damage" to the network by injecting data. The overall damage can be expressed in terms of "congestion costs" and is usually a function of integrated excess input rate or queue size at the network nodes. Without packet loss, a user's contribution to the congestion costs is entirely *external* to the user. He does not have to pay for it, and will not even notice it until congestion is so high that packet loss cannot be avoided. It is therefore necessary to *internalize* these costs. If all users "pay" for the "damage" they do to the network, they will take the costs into account when deciding how much bandwidth they use. This price is the so-called *Shadow Price*.

These *Shadow Prices* are part of the distributed game and can be viewed as taxes. This is also the solution to the aforementioned pollution problem. If a tax is imposed on every polluter reflecting the increase of damage to the environment (Shadow Price), the costs are internalized. The polluter can still freely choose how much he pollutes the environment, but he is obliged now to take the fee into consideration. Thus, he will in his own self-interest optimize the benefit from polluting (for example by being able to produce profitable goods) minus the costs that are attached to it. Let us assume that the pollution will cause smog, which is not desired by any participant. Furthermore, assume that the taxes are designed in such a way that the government wants to maximize profits under the condition that no smog occurs. If risk of smog is high, taxes will be high to reduce pollution. If, on the other hand, the risk of smog is very low, the government will decrease taxes. Then more people will decide to produce goods and cause pollution, and – under certain mathematical conditions – overall tax income will grow. This form of market game between polluters and government will lead to optimal use of resources, i.e. optimal tax income, optimal production, and no smog. People who are willing to spend more money can produce more, causing more pollution, and run a higher risk of generating smog. This is the second advantage of the market game as it allows for *service differentiation* depending on the budget or *willingness to pay* of each participant.

The same applies to a network: it should be used optimally but not overloaded. When the network is empty, "congestion taxes" are low and users will use more bandwidth than when the network is busy and "congestion taxes" are high. As will be shown mathematically in the following section, this market game will lead to an optimal solution of the bandwidth sharing problem, the so called Nash bargaining solution. This desirable property of the market game has motivated a growing body of research [MMV95,Kel97,KMT98,GK99,Key01,LL99,AL00, Low00].

One could think of a network design where customers actually pay those "congestion taxes". Real money is an incentive for the users to play the game correctly. However, real money is not necessary for the game to work. One could think of a fictitious currency that is part of a service level agreement (SLA) between network operator and user. Or, the game is just viewed as a solution to the optimization problem and it is just assumed that all users follow the rules. This is not unrealistic if the rules of the game are hidden in the TCP stack so that only experts could manipulate them. The *Explicit Congestion Notification* (ECN) proposal, which is already deployed on the Internet, also allows manipulation. Even conventional TCP stacks can be manipulated such that the user receives higher data rates at the cost of everyone else. However, until now such manipulations have not been a measurable problem. It remains an open question if TCP stack manipulation will become a problem on the Internet in the future. If, alternatively, real money is used, what happens to the money, considering it was earned by the network operator in a situation where the network's capacity was not large enough? It seems contradictory that money is earned because the capacity was too small. This problem could for example be solved by refunding this money equally to the customers. Furthermore, users like to know in advance what network usage will cost. For these reasons, the usage of virtual currencies is suggested. The virtual currency can still be part of a service level agreement that is paid for in real money. Also, the network provider could install policers that enforce “fair-play”.

The great advantage of this market game is that it can easily be implemented for a network. The decision making, i.e. the intelligence, is placed in the end hosts of the network. The network has the only to generate the tax signals. Since these signals only depend on the congestion state and not on single flows, no flow state information has to be kept at the network nodes. This allows for simple routers and good scalability – a great advantage for the Internet. Additionally, since the decision making happens at the end hosts, users can freely decide how they spend their money. There is no need for the network to know an application’s bandwidth demand profile or to recognize users that should receive a higher service level than the other others. All this is taken care of at the end systems. Thus, Congestion Pricing allows for service differentiation that is much simpler than the Differentiated Services (DiffServ) [BBC$^+$98] or the Integrated Services (IntServ) [BCS94] approach by the IETF. The implementation issues will be described more in detail after the introduction of the mathematical model.

As mentioned before, Congestion Pricing does not imply charging of real money. However, many people associate money with the term “Congestion Pricing” and therefore object to this idea. Therefore the more positive term “Resource Marking” will be used synonymously in this dissertation. *Resource Marking* implies that Congestion Pricing theory can be applied without actually charging anything. Again, *Resource Marking* and *Congestion Pricing* refer to the same mathematical model, but their connotation is different.

3.2 Mathematical Model and Important Properties

3.2.1 Basic Model Without Delays

A network and its resources can be viewed as a set of links $\mathcal{L}$. Attached to the network is a set $\mathcal{N}$ of source–sink pairs. Suppose that each source $n \in \mathcal{N}$ is associated with a user, and that user has a utility $U_n(x_n)$ when the source is sending at rate x_n. This utility function characterizes the user’s preference for bandwidth and is assumed to be an increasing, strictly concave, and continuously differentiable function for $x_n > 0$[1]. This is true for *elastic* traffic (cf. Section 2.2). An example utility function is depicted in Figure 3.2.1. The utilities are assumed to be additive, therefore the aggregate utility of rate allocation $\mathbf{x} = (x_n, n \in \mathcal{N})$ can be written as $\sum_{n\in\mathcal{N}} U_n(x_n)$. The flow of data from each source-sink pair consumes resources along a path through the network. These paths consist of links $l \in \mathcal{L}$. A 0–1 incidence matrix $\mathbf{A} = (A_{ln}, l \in \mathcal{L}, n \in \mathcal{N}, A_{ln} \in \{0,1\})$ indicates whether a link l is used by source n or not. Each link l has a maximum capacity c_l. Let further be $\mathbf{c} = (c_l, l \in \mathcal{L})$.

[1]Inactive sources are eliminated from the set $\mathcal{N}$.

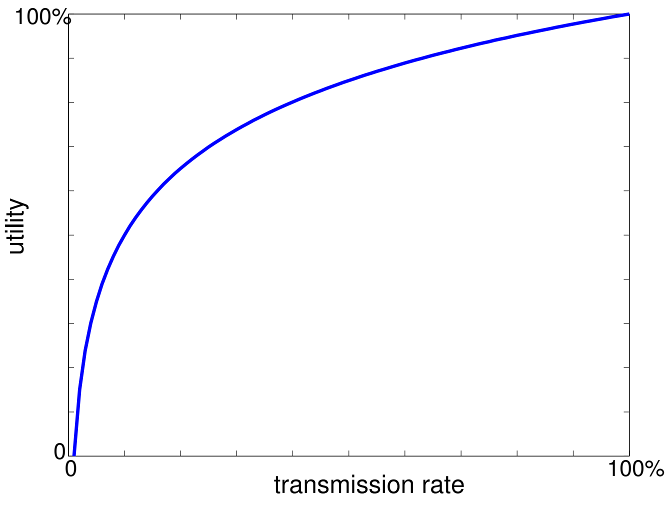


Figure 3.2.1: Utility function for elastic traffic

Global Optimization Problem

The optimization problem for the entire system to globally find the optimal rates is then:

$$\text{maximize} \quad \sum_{n\in\mathcal{N}} U_n(x_n) \tag{3.2.1}$$

$$\text{under the constraint} \quad \mathbf{Ax} \leq \mathbf{c} \tag{3.2.2}$$

$$\text{over} \quad \mathbf{x} > \mathbf{0} \tag{3.2.3}$$

The constraint says that the total of all rates through a link cannot be greater than its capacity. In [Kel97], a solution using Lagrangian methods is presented, which is shown here in a slightly simplified form:

$$L(x,y,\lambda) = \sum_{n\in\mathcal{N}} U_n(x_n) + \lambda^T(\mathbf{c} - \mathbf{Ax} - \mathbf{z}), \tag{3.2.4}$$

where $\lambda = (\lambda_l, l \in \mathcal{L})$ is a vector of Lagrangian multipliers and $\mathbf{z} = (z_l, l \in \mathcal{L})$ is a vector of positive slack variables. Then

$$\frac{\partial L}{\partial x_n} = U_n'(x_n) - \sum_{l\in\mathcal{L}} A_{ln}\lambda_l$$

For $\mathbf{x} > \mathbf{0}$, $\lambda \geq \mathbf{0}$, and $\lambda^T(\mathbf{c} - \mathbf{Ax}) = \mathbf{0}$ at the optimum, the following first-order condition must be satisfied [BSMM99]:

$$U_n'(x_n) = \sum_{l\in\mathcal{L}} A_{ln}\lambda_l \tag{3.2.5}$$

The interpretation of the Lagrange multipliers λ_l as marginal costs or *shadow prices*[2] of additional capacity at link l leads to a different view of the problem: The charge per rate or *congestion price* p_n for each source n is then given as:

$$p_n = \sum_{l\in\mathcal{L}} A_{ln}\lambda_l \tag{3.2.6}$$

[2]The shadow price has the unit *cost*, the "cost" could be anything, e.g. lost packets, delay or money. Here the unit 1 is used.

Hence, at the social optimum, the derivative of a user's utility function exactly matches the sum of the shadow prices of all resources along the user's route:

$$U_n'(x_n) = p_n. \tag{3.2.7}$$

Distributed Optimization Problem

As described in the introduction, the global optimization problem requires knowledge of all users' utility functions. This is not mathematically tractable for larger networks. F. Kelly has suggested considering two simpler optimization problems: The user problem and the network problem. If the *price* p_n is "charged" to the user per rate, and if the user is allowed to freely vary its rate x_n, the user will try to solve the following optimization problem:

$$\begin{aligned} \text{maximize} \quad & U_n(x_n) - p_n x_n && (3.2.8)\\ \text{over} \quad & x_n > 0 && (3.2.9) \end{aligned}$$

The network, on the other hand, will try to optimize its revenue:

$$\begin{aligned} \text{maximize} \quad & \sum_{n \in \mathcal{N}} p_n x_n && (3.2.10)\\ \text{under the constraint} \quad & \mathbf{Ax} \leq \mathbf{c} && (3.2.11)\\ \text{over} \quad & \mathbf{x} > \mathbf{0} && (3.2.12) \end{aligned}$$

F. Kelly proves in [Kel97] the existence of a price vector $\mathbf{p} = (p_n, n \in \mathcal{N})$ such that the vector $\mathbf{x} = (x_n, n \in \mathcal{N})$ of unique solutions to the user problems (3.2.8)–(3.2.9) also solves the network problem (3.2.10)–(3.2.12). In this case, the vector $\mathbf{x}$ also solves the system's optimization problem (3.2.1)–(3.2.3).

3.2.2 Relaxed Model (Penalty Approach)

In the system problem, constraint (3.2.2) implies that the total of all rates through a link cannot be greater than its capacity. Consider a network that has buffer space at the network nodes. In such a network, rates can temporarily exceed the capacity of the link. However, if the buffer fills up, there is an additional delay or — even worse — packet loss. So the constraint (3.2.2) should be replaced by a *penalty* or *cost function* $C_l(y_l)$ that is assumed to be convex and describes the undesired increase in delay or packet loss probability when link $l \in \mathcal{L}$ has a load y_l. The *relaxation* of the system problem is then [GK99]:

$$\begin{aligned} \text{maximize} \quad & \sum_{n \in \mathcal{N}} U_n(x_n) - \sum_{l \in \mathcal{L}} C_l\Big(\sum_{n \in \mathcal{N}} A_{ln} x_n\Big) && (3.2.13)\\ \text{over} \quad & \mathbf{x} > \mathbf{0} && (3.2.14) \end{aligned}$$

This leads to the identification of the shadow prices with the derivative of the cost function:

$$\lambda_l(y_l) = C_l'(y_l)$$

Since $C_l(y_l)$ is assumed to be a convex function, $\lambda_l(y_l)$ is greater than zero and increasing.

Thus the equation

$$p_n = \sum_{l \in \mathcal{L}} A_{ln} \lambda_l(y_l) = \sum_{l \in \mathcal{L}} A_{ln} C_l'(y_l) \tag{3.2.15}$$

describes, how the path price is determined from the cost function (“path price algorithm”).

3.2.3 User’s Rate Adaptation

Putting everything together, a rule for each user to update his rate can be found that will lead to convergence to the optimal solution x_n^*. F. Kelly proposes the following *rate differential equation* [KMT98]:

$$\frac{d}{dt} x_n(t) = \kappa_n (x_n(t) U_n'(x_n(t)) - p_n(t) x_n(t)) \tag{3.2.16}$$

with the strictly positive gain $\kappa_n > 0$. In the remainder of this dissertation, such a user’s rate adaptation rule will be referred to as “source algorithm”. A system using this adaptation rule has been proven to converge to the optimum and to be stable, but instantaneous price updates are assumed. It is obvious from (3.2.16) that in optimum

$$\frac{d}{dt} x_n(t) = 0 \Longrightarrow U_n'(x_n^*) = p_n^*,$$

which is identical to requirement (3.2.7). Note that this algorithm is independent from the actual shadow price calculation, which is derived from the cost function $C_l(y_l)$. The differential equation (3.2.16) is designed such that the transmission rate $x_n(t)$ will converge to the optimal value x_n^*.

3.2.4 Stability and Convergence with Delays

So far, price updates were assumed to be instantaneous. In a real network, however, there will be a delay until a source receives pricing information. Stability and convergence of the system in the case of delays were examined by L. Massoulié [Mas00]. He has proven based upon work by Johari and Tan [JT01] that for delayed price updates the system is stable if:

$$\overline{\kappa}_n (p_n + \sum_{l \in \mathcal{L}} A_{ln} \lambda_l' y_l^*) < 1, \tag{3.2.17}$$

where $\mathbf{y}^*$ is the equilibrium vector of loads: $y_l^* = \sum_{n \in \mathcal{N}} A_{ln} x_n^*$; and $\overline{\kappa}_n := \kappa_n RTT_n$, where κ_n is the gain used in (3.2.16).

More general and detailed mathematical proofs for stability in the case of delays and for specific implementations can also be found in [YL01, IB01, WP02, LWG03]. They are omitted here, but stability and convergence speed of actual implementations will be a re-occurring concern in the following chapters. In Chapter 6, stability will be examined using control theory.

3.2.5 Duality Model and Gradient Projection Method

An alternative solution to the optimization problem (3.2.1)–(3.2.3) using *duality theory* was presented by S. Low and D. Lapsley in [LL99]. While the solution to the original problem is a vector of rates $\mathbf{x}^*$, in the dual problem, the vector of shadow prices λ^* is required. Thus rates $x_n(t)$ are regarded as primal variables and shadow prices $\lambda_l(t)$ as dual variables. The dual problem to the primal problem (3.2.1)–(3.2.3) is:

$$\text{minimize}_{\mathbf{p}\geq\mathbf{0}} \quad D(\mathbf{p}) := \sum_{n\in\mathcal{N}} B_n(p_n) + \sum_{l\in\mathcal{L}} \lambda_l c_l, \tag{3.2.18}$$

where

$$B_n(p_n) = \max_{x_n>0} U_n(x_n) - p_n x_n. \tag{3.2.19}$$

S. Low and D. Lapsley proposed to use the *gradient projection method* by adjusting the link prices to solve the dual problem:

$$\lambda_l(t+1) = \left[\lambda_l(t) + \gamma\frac{\partial D}{\partial\lambda_l}(\mathbf{p}(t))\right]^+,$$

where $\gamma > 0$ is a step size, and $[x]^+ := \max(0,x)$. As shown in [LL99],

$$\frac{\partial D}{\partial\lambda_l}(\mathbf{p}) = c_l - y_l(\mathbf{p}),$$

which is used to derive the shadow price update rule for link l:

$$\lambda_l(t+1) = [\lambda_l(t) + \gamma(y_l(\mathbf{p}(t)) - c_l)]^+. \tag{3.2.20}$$

Thus, in this case the calculation of the shadow prices is not derived from a cost function $C_l(y_l)$ as in Subsection 3.2.2, but from the gradient projection method that can be used to solve the dual problem (3.2.18)–(3.2.19). The *Random Exponential Marking (REM)* method that will be introduced in Section 5.2 was designed using this approach.

3.2.6 Logarithmic Utility Functions and Proportional Fairness

Suppose that each user's utility function is of the form

$$U_n(x_n) = w_n \ln(x_n) \tag{3.2.21}$$

where $w_n \geq 0$ is a scalar that is often called weight or *willingness to pay*[3]. This class of user's utility functions (3.2.21) has the advantage that it leads to a *proportionally fair* rate allocation at the social optimum as well. This was proven in [Kel97, KMT98]. A proportionally fair rate allocation maximizes the sum of all logarithmic utility functions. Take, for example, three users (`A`,`B`,`C`) having identical utility functions. Also consider a double bottleneck link topology (Figure 3.2.2), where the two core links have the same capacity. If user `A` uses only the first link of the network, user `B` only the second, and user `C` uses both links, a proportionally fair rate allocation will then give each of the first two users a share of α times a link's capacity, and user

[3]Unit $rate\,of\,budget$, here $\frac{pkts}{s}$

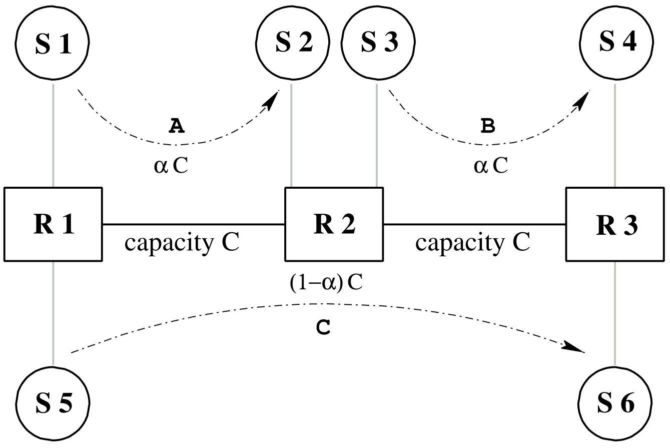


Figure 3.2.2: Double bottleneck link topology

C a share of $(1-\alpha)$ times a link's capacity, where $\alpha = \frac{2}{3}$ is established solving (3.2.1)–(3.2.3):

$$\begin{aligned}
\text{maximize} \quad & \sum_{n=1,2,3} U(x_n^*) = 2w\ln(\alpha C) + w\ln((1-\alpha)C) \\
\text{under the constraint} \quad & \alpha < 1 \\
\text{over} \quad & \alpha > 0 \\
\Longrightarrow \quad & \frac{2}{\alpha} - \frac{1}{(1-\alpha)} \overset{!}{=} 0 \\
\Longrightarrow \quad & \alpha = \frac{2}{3}.
\end{aligned}$$

Note that this allocation is different from *max-min fairness*, where each user would receive half of a link's capacity. Also note that a proportionally fair rate allocation is independent of the connections' round-trip delays, which is in contrast to conventional TCP.

A more formal definition was given by F. Kelly and M. Biddiscombe [Bid]. A rate allocation $\mathbf{x}$ is *proportionally fair* if it is feasible and if for any other feasible vector $\mathbf{x}^*$ the aggregate proportion of changes is non-positive:

$$\sum_{n=1}^{N} \frac{x_n^* - x_n}{x_n} \leq 0. \tag{3.2.22}$$

Such a vector $\mathbf{x}$ is also the Nash bargaining solution. By choosing the willingness to pay w_n, one can give weights to each connection. The rates are then *weighted proportionally fair* [LA00] or equivalently the *rates per unit charge are proportionally fair* [KMT98]:

$$\sum_{n=1}^{N} w_n \frac{x_n^* - x_n}{x_n} \leq 0. \tag{3.2.23}$$

3.3 Conclusions

In this chapter the mathematical model of Congestion Pricing theory was presented. Mathematically, Congestion Pricing is a distributed optimization problem. If solved, the global optimization problem will also be solved. Thus, Congestion Pricing seems to be ideal for application on a network where distributed intelligent sources are connected to a network of relatively simple core elements. Mathematically, such a network would always operate optimally.

However, in a real network information cannot be transmitted instantaneously. Delays in the transport of shadow prices could lead to stability problems [Mas00]. Further, networks are usually packet–oriented. They do not use a continuous transmission rate, instead they use a transmission rate of packets or even congestion windows such as TCP. Thus, the problem is discrete. Also protocols have to be developed that allow the transportation of the shadow prices to the sources. This leads to compatibility issues if such a protocol should operate on a network that already exists — such as the Internet. These are the main problems that this dissertation focuses on.

Chapter 4

Implementation of Congestion Pricing Based TCP

In the previous chapter the mathematical framework of Congestion Pricing theory was introduced. Now in this chapter a TCP implementation that makes full use of Congestion Pricing theory will be presented. It is a direct adaptation of the theory to packet based networks and TCP's congestion window principle, using full pricing information. Although this algorithm is based on TCP, it requires significant changes to the network and to both TCP sender and receiver. It will show that these modifications are honored with superior performance.

4.1 Implementation Issues

So far some important properties of Congestion Pricing theory were presented. When it comes to implementation, however, three important questions still have to be answered:

1. How do sources update their rates? ("source algorithm")

2. How do network nodes calculate the current shadow prices? ("link algorithm")

3. How is the path price calculated and transmitted to the sender? ("path price transport")

For example, because real networks are packet based, rate updates of all users are not synchronous and continuous, but asynchronous and discrete. In the next section these three issues will be addressed whereby a direct adaptation of the algorithms that were presented in Chapter 3, is developed. The resulting TCP variant will then be called *Congestion Pricing-TCP with Explicit Price Feedback (CP-TCP/EPF)*. The Explicit Price Feedback (EPF) method with direct congestion window updates is the most straightforward adaptation of Congestion Pricing theory to TCP. All network nodes will calculate congestion costs and add the shadow prices to an options field in the IP header. The receiver then returns the sum of all shadow prices, the path price, to the sender, which will then interpret this information to update its congestion window. The performance of the implementation will be evaluated by means of simulation.

4.2 Congestion Pricing-TCP with Explicit Price Feedback (CP-TCP/EPF)

4.2.1 Source Algorithm (Direct Window Update Algorithm)

The source algorithm is based on F. Kelly's rate differential equation (3.2.16). Further, a logarithmic utility function of the type $U_n(x_n) = w_n \ln(x_n)$ is assumed, where w_n is user n's *willingness to pay*. As will be shown later, the willingness to pay can be used to give priorities to certain users. Since $U_n'(x_n) = \frac{w_n}{x_n}$, the rate differential equation (3.2.16) can be translated to a *rate update rule*:

$$x_n(t+T) := x_n(t) + \kappa_n T \left(w_n - p_n(t)x_n(t)\right), \tag{4.2.1}$$

where T is the time between two updates, and κ_n is the gain per update interval.

Additionally, this rule must be modified to work with window based algorithms such as TCP. The congestion window (*cwnd*) controls the amount of data that can be injected into the network before it is acknowledged by the receiver. Instead of rate updates, the source algorithm has to update the size of the congestion window. If the assumption is made such that the round-trip time (RTT) is approximately constant for small time intervals, the following approximation can be made:

$$\frac{\partial(cwnd)}{\partial t} \cong RTT \cdot \frac{\partial x_n}{\partial t}.$$

Insertion of this formula in (4.2.1) yields:

$$cwnd_n(t+T) := cwnd_n(t) + \kappa_n T \cdot RTT_n \cdot \left(w_n - p_n(t)x_n(t)\right).$$

This source algorithm assumes rate adjustments every T seconds. Without timeouts, TCP only adjusts its congestion window when an acknowledgment is received, which happens approximately at a rate of $\frac{\underline{cwnd}}{RTT}$[1]. The update rate has to be taken into consideration when calculating the size of the window adjustment. Thus:

$$T(t) = \frac{RTT(t)}{\underline{cwnd}(t)}.$$

The source algorithm is then modified accordingly:

$$\begin{aligned}\Delta cwnd_n(t) &= \kappa_n \cdot \frac{RTT_n(t)}{\underline{cwnd_n}(t)} \cdot RTT_n(t) \cdot \left(w_n - p_n(t) \cdot \frac{cwnd_n(t)}{RTT_n(t)}\right) && (4.2.2)\\ &\cong \bar{\kappa}_n \left(w_n \cdot \frac{RTT_n(t)}{cwnd_n(t)} - p_n(t)\right), && (4.2.3)\end{aligned}$$

where κ is the gain per update interval; and $\bar{\kappa}_n = \kappa_n \cdot \overline{RTT_n}$ is defined as gain. $p_n(t)$ is the path price as seen by source n at time t. Units are shown in Table 4.2.1. Since p_n can take any value and the amount of congestion window size change directly depends on the value of p_n, this source algorithm will be called *Direct Window Update Algorithm.*

[1]Note that this is a window update rate, units are thus $\frac{1}{s}$, not $\frac{pkts}{s}$. $\underline{cwnd}$ is used to denote the different unit 1. In the later sections units will be ignored and the numerically identical *cwnd* will be used only. In actual implementations, *cwnd* is usually measured in bytes instead of packets, thus $\underline{cwnd}$ and *cwnd* are also numerically different.

Table 4.2.1: Units of variables in source algorithm

Variable	Unit
$cwnd_n$	$pkts$
$\underline{cwnd_n}$	1
κ	$\frac{1}{s}$
$\overline{\kappa}$	1
RTT_n	s
w_n	$\frac{pkts}{s}$
p_n	1

4.2.2 Link Algorithm

Several link algorithms have been proposed. They depend on the particular cost function being considered. Some of them will be presented and evaluated in Subsection 5.4.4. Here, a link algorithm that has been proposed by S. Athuraliya and S. Low will be used. For their *Random Exponential Marking (REM)* [AL00] they suggested three different *shadow price computation* rules (PC1, PC2, and PC3) of which they identified PC3 as the most promising one:

$$\lambda_l(t+1) = [\lambda_l(t) + \gamma(\alpha_l(b_l(t) - b_0) + \widehat{y}_l(t) - c_l)]^+, \tag{4.2.4}$$

where c_l is the capacity of the link, b_0 is the desired equilibrium queue size, $\alpha_l > 0$ is a small constant that weights the influence of the instantaneous queue size on the shadow price, and $\gamma > 0$ is the step-size, which must be sufficiently small to ensure convergence to the unique optimal rates [LL99]. This algorithm was derived from the shadow price update rule (3.2.20). Shadow prices must be non-negative as indicated by $[]^+$. $\widehat{y}_l(t)$ is the link's estimated aggregate input rate in average sized packets per sample interval at time t, given by:

$$\widehat{y}_l(t+T) = (1-\delta)\widehat{y}_l(t) + \delta\left(\frac{bytes/mean\_packet\_size}{T}\right), \tag{4.2.5}$$

where T is the sample interval. In the following simulations, a sample interval of 1 ms and weight $\delta = 0.9$ will be used.

4.2.3 Path Price Transport

Finally, all shadow prices on the path have to be added up and transported to the sender. Here, a fictitious new IP options field[2] is used. Each router then adds the local shadow price to the old value in that field. The receiver of the IP packet reads this value and echoes it back to the sender in a TCP options header field. Thus, two new options have to be introduced to TCP/IP. Since pricing information is completely transmitted, this type of implementation is referred to as *Explicit Price Feedback.* The advantage of Explicit Price Feedback is that no information is lost. This allows stronger reactions by the sender when the price has changed significantly, and

[2] An actual implementation could also use the 16-bit identification field when the Don't Fragment (DF) flag bit is 1. With newer TCP implementations, this is commonly the case.

smaller adjustments of the sending rate when the price difference is small. However, both TCP and IP have to be modified to carry the pricing information.

4.3 Performance Evaluation

4.3.1 Simulation Setup (Double Bottleneck Link Network)

By means of simulation, the superior performance of Congestion Pricing based TCP with Explicit Price Feedback ("*CP-TCP/EPF*") over conventional TCP variants such as TCP Tahoe, TCP Reno, and TCP NewReno is demonstrated. Since the congestion avoidance algorithms shall be compared without the influence of retransmissions and timeouts, the simulation setup is designed in such a way that packet loss can completely be avoided. For this reason, CP-TCP/EPF is compared to a "state-of-the-art" TCP NewReno implementation that uses the ECN extensions (cf. Subsection 2.4.2) and RED as the Active Queue Management strategy (cf. Subsection 2.4.3). The performance of TCP NewReno with a conventional drop-tail queue that is most commonly in use at present will also be shown. The simulation setup described here is used throughout the dissertation to compare different algorithms and approaches.

Simulations were performed using the UCB/VINT *Network Simulator 2 (ns-2)* [UCB]. Two simulation scenarios are used to focus on two different aspects of performance. The first scenario "Rate Allocation" is used to evaluate the capability of the TCP variants to establish a *weighted proportionally fair* rate allocation at the steady-state. The second experiment "Dynamics" focuses on the dynamic behavior of the algorithms when network conditions change. In the "Dynamics" scenario, additional flows are turned on and off at certain times during the simulation. Both simulations make use of a "double bottleneck link" topology (cf. Figure 4.3.1). It consists of two bottleneck links with a bandwidth of 96 Mbps each. All other links

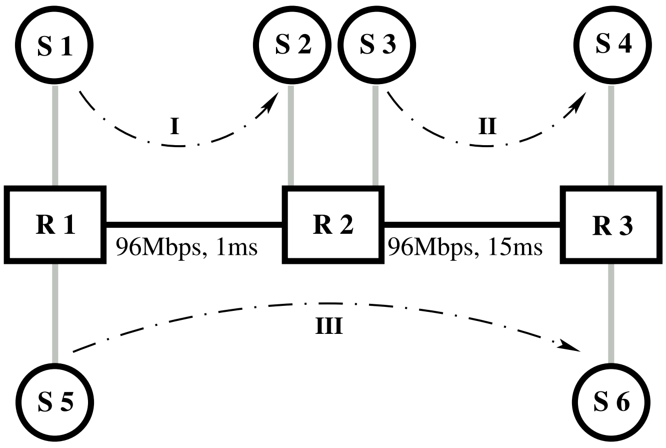


Figure 4.3.1: Double bottleneck link topology

have a bandwidth of 1000 Mbps and are therefore not limiting the rates. For the performance analysis, the positions of the bottleneck links are not significant. Thus, this topology is a realistic representation of the Internet, where usually the access link and at most one other link are limiting the transmission rate.

The delay of all links is 1 ms except for the second core link, which has a delay of 15 ms. This allows the evaluation of the influence of different round-trip times. The three paths on the network are denoted as path I through path III. TCP's congestion avoidance algorithm as implemented in ns-2 was modified to reflect the final source algorithm (4.2.3). Slow start and TCP's normal reaction to packet loss were left unchanged. Similarly, the link algorithms and packet classes were replaced to implement the explicit price feedback. Table 4.3.1 shows the chosen parameters for the simulations.

Table 4.3.1: Link algorithm parameters

Algorithm	Parameters
CP-TCP/EPF	$\phi = 1.06, \gamma = 0.001, \alpha = 0.1, b_0 = 2$
RED	$\overline{\kappa} = 0.01, \gamma = 0.8305, b_0 = 2$

First Simulation: Rate Allocation

Each path carries 12 flows from greedy sources with different *willingness to pay* parameters w_n (cf. Table 4.3.2). Since TCP NewReno does not use a *willingness to pay* parameter, it is omitted

Table 4.3.2: Simulation setup

Flow id	Path	# Bottleneck links	RTT [ms]	w_n
1 - 4	I	1	6	100
5 - 8	I	1	6	200
9 - 12	I	1	6	400
13 - 16	II	1	34	100
17 - 20	II	1	34	200
21 - 24	II	1	34	400
25 - 28	III	2	36	100
29 - 32	III	2	36	200
33 - 36	III	2	36	400

for conventional TCP. A proportionally fair rate allocation will allocate $\frac{2}{3}$ of the capacity of the first link to the connections using path I, and $\frac{1}{3}$ to the connections using path III, as was described in Subsection 3.2.6. The rate allocation for the second link is analogously, $\frac{2}{3}$ of the capacity to connections using path II, and $\frac{1}{3}$ to the connections using path III.

All connections were started with a congestion window of one at time zero. The number of packets each connection could transfer during the following 60 seconds was then measured to determine throughput. Since packet losses were avoided, no packets had to be retransmitted. Goodput and throughput are therefore identical.

Second Simulation: Dynamics

The second simulation scenario is used to examine the ability of the algorithms to cope with changing network conditions. For the first 20 seconds, the simulation setup is identical to the first simulation. However, after 20 seconds the number of flows per path is doubled. Then every 20 seconds the network conditions change again (cf. Figure 4.3.2). In order to avoid

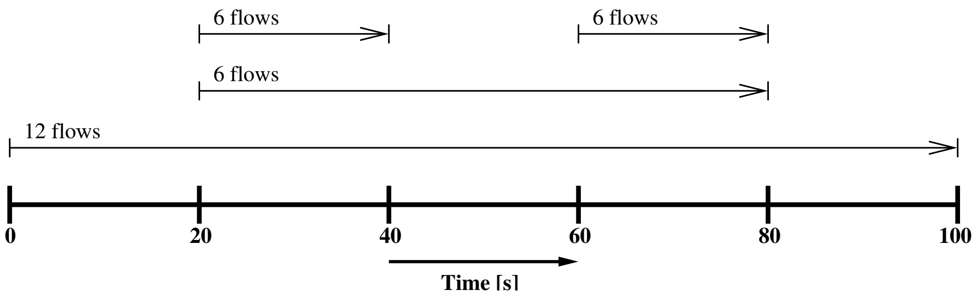


Figure 4.3.2: Start and stop times of flows per path

under-utilization of the bottleneck links after some of the sources are turned off, and to avoid additional queuing delay and congestion when additional sources are turned on, the congestion windows should be adapted inversely to the changes in load: From (2.3.1)

$$cwnd_n^* = x_n^* \cdot RTT_n^*,$$

where the available transmission rate x_n^* is inversely proportional to the load factor f, which determines the number of active sources $N = f \cdot (N_1 + N_2 + N_3)$:

$$x_n^* = \frac{1}{f} \frac{w_n}{N_1 w_1 + N_2 w_2 + N_3 w_3} c_l.$$

This is the result from the use of a logarithmic utility function (cf. Subsection 3.2.6) with three different *willingness to pay* values w_1, w_2, w_3.

Figure 4.3.3 displays such an idealized progression of the congestion window. In an actual network, there will always be a delay until the changes in load are fed back to the sources, thus the progression of the congestion windows will differ and temporary under-utilization and congestion cannot completely be avoided. Nonetheless, the tendency should be similar to the ideal case.

4.3.2 Simulation Results

Rate Allocation

The resulting rate allocation is shown in Figure 4.3.4 for CP-TCP with Explicit Price Feedback. In comparison, the resulting rate allocations of TCP NewReno with drop-tail queues and RED active queue management are given in Figure 4.3.5.

The horizontal lines indicate the rates that correspond to a *weighted proportionally fair* rate allocation. Since conventional TCP implementations such as NewReno do not achieve a weighted proportionally fair rate allocation, the lines are only relevant to the Congestion Pricing based TCP implementations. The same plot types will be used later for other TCP variants.

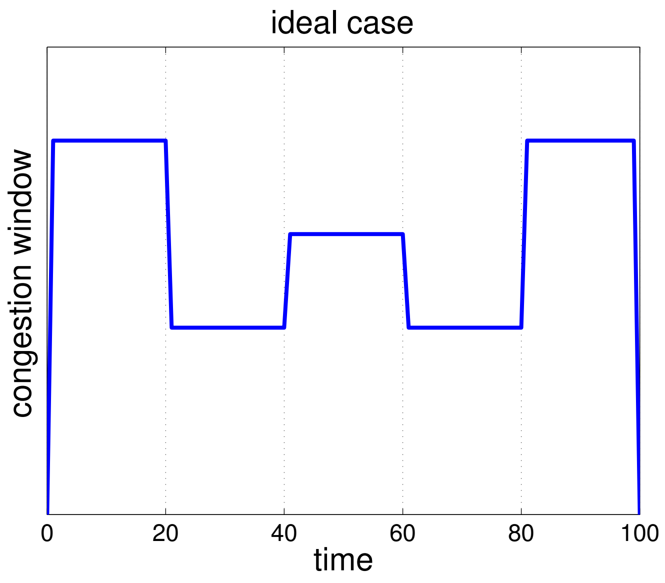


Figure 4.3.3: Idealized progression of the congestion window

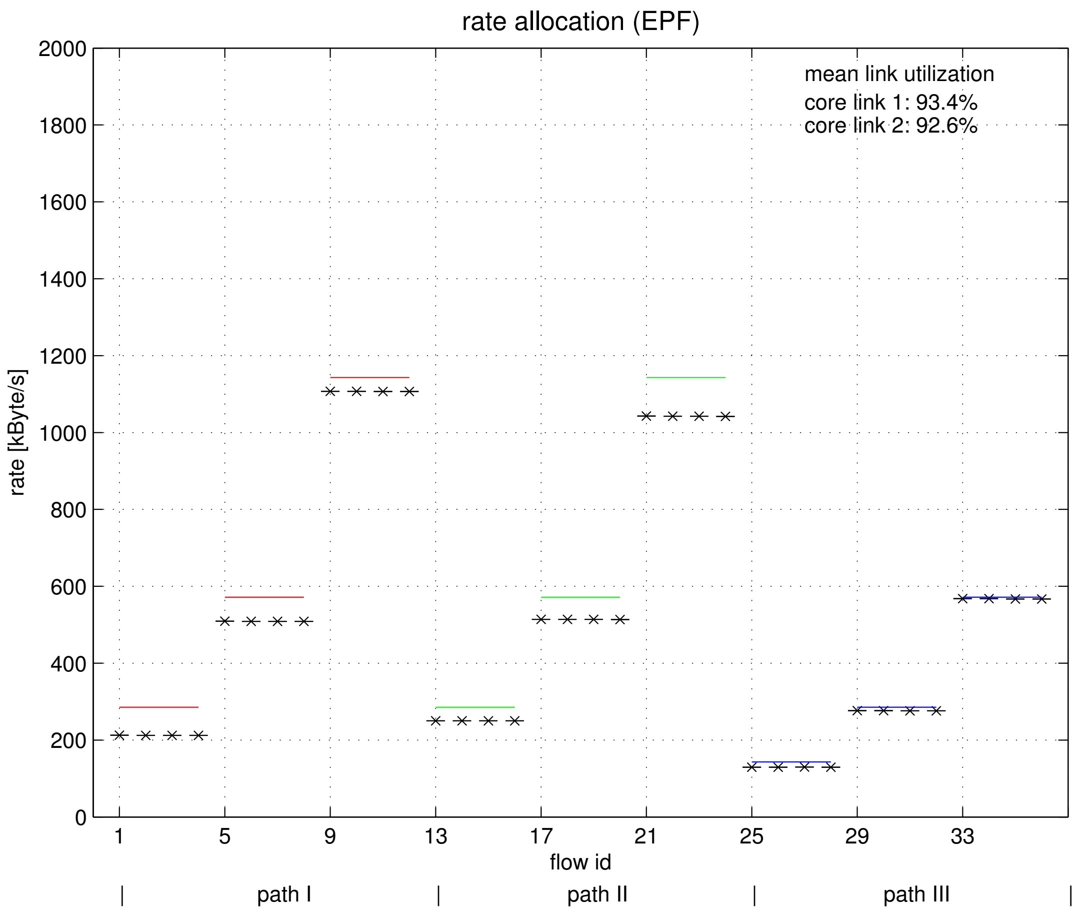
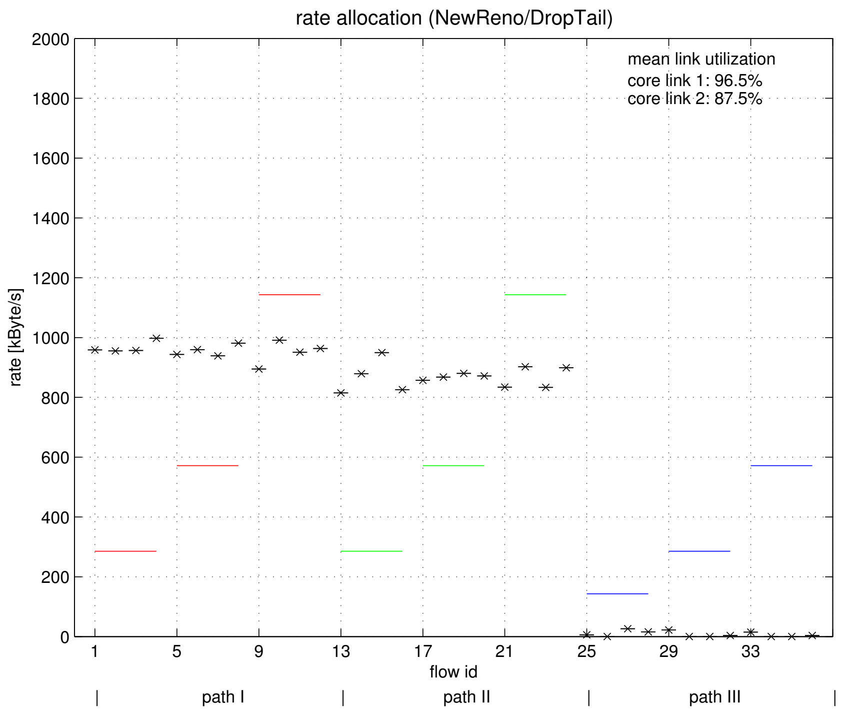
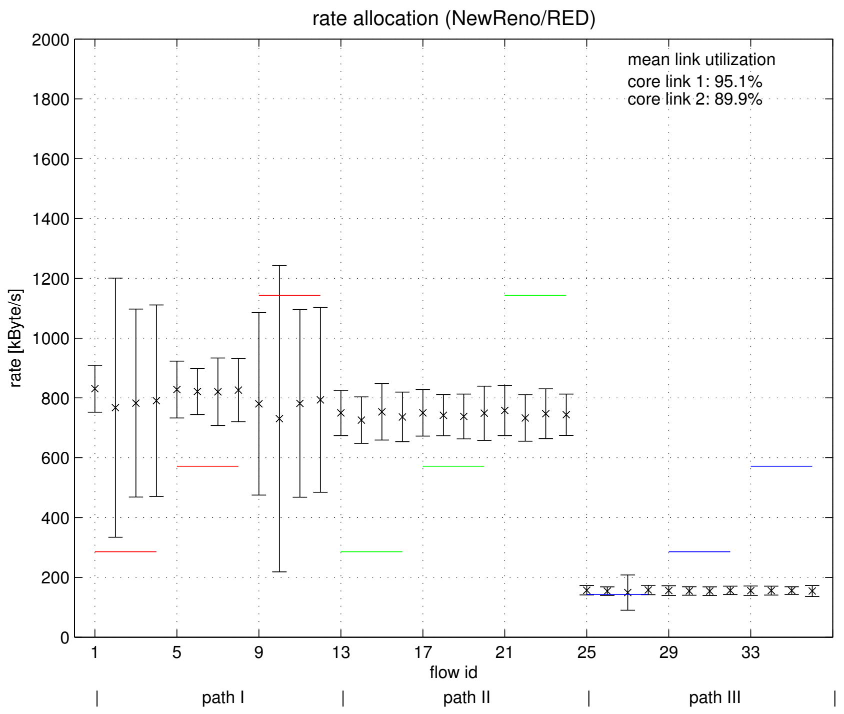


Figure 4.3.4: Rate allocation of Congestion Pricing TCP [Zim03]



(a) TCP NewReno/Droptail



(b) TCP NewReno/RED

Figure 4.3.5: Rate allocation of conventional TCP (NewReno with drop-tail and RED queues) [Zim03]

Optimally, the actual rate allocation shown by the cross marks should match the theoretical values indicated by the horizontal lines. For the longest path III, there is a nearly perfect match (cf. Figure 4.3.4). But even for the other two paths, the Congestion Pricing based TCP variant nearly achieves a *weighted proportionally fair* rate allocation. It is not surprising that in total the actual rate is slightly lower than the theoretical values because the theoretical values represent perfect 100% utilization all the time, which is not achievable in a real network with feedback delays. The achieved bottleneck utilization for all three variants are shown in Table 4.3.3.

Table 4.3.3: Bottleneck link utilization

TCP variant	Utilization of core link 1	Utilization of core link 2	Average core link utilization
TCP NewReno+drop-tail	96.5 %	87.5 %	92.0 %
TCP NewReno+RED	95.1 %	89.9 %	92.5 %
CP-TCP/EPF	93.4 %	92.6 %	93.0 %

Note that for example the rates assigned to the sources on path I and II only depend on the *willingness to pay*, but not on the delay assigned to the links. This is different with TCP NewReno. The sources on path II received a lower bandwidth share than the sources on path I (cf. Figure 4.3.5a). Furthermore, while for CP-TCP/EPF all four flows of the same source receive nearly identical bandwidth shares, there are significant differences for TCP NewReno as well as with drop-tail queues and RED queues. Thus, fairness between identical flows can only be achieved with Congestion Pricing.

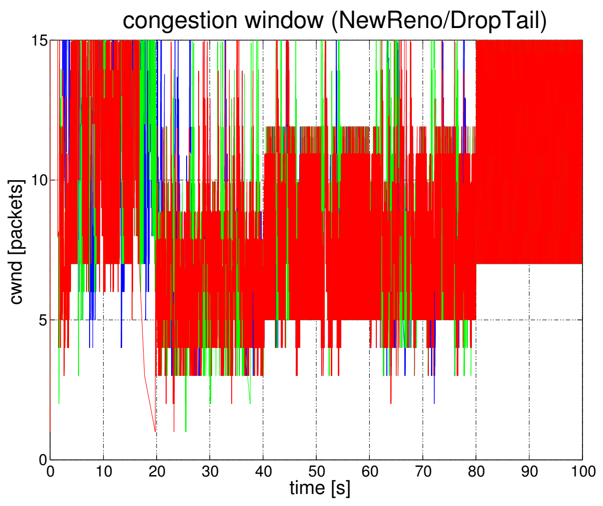
Included in the figures are also 95% confidence intervals for the resulting rate allocation. Since TCP NewReno with drop-tail queues and CP-TCP/EPF work entirely deterministically, for every run the same rate allocation will result. This is different for TCP NewReno with RED, which introduces a random component and thereby changes rate allocations. The large confidence intervals shown in Figure 4.3.5b also indicate that the rate allocation is somewhat random leading to unfair rate distributions.

These rate allocation experiments have shown that CP-TCP with Explicit Price Feedback yields the desired *weighted proportionally fair* rate allocation nearly perfectly. The *willingness to pay* parameter can be used to implement relative Quality of Service (QoS) without the need for central policers or bandwidth reservation. Resulting rate allocations closely match the theoretical values.

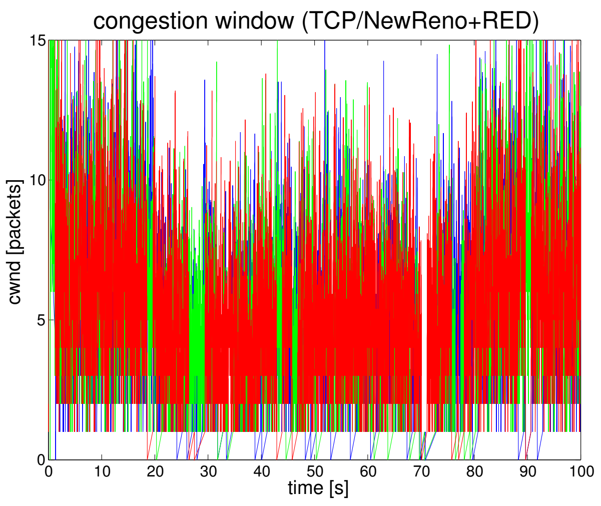
Dynamics

Additional to the rate allocation, it is important how the TCP algorithm behaves over time. The dynamics of the congestion window for CP-TCP/EPF in comparison to TCP NewReno with drop-tail and RED queues are shown in Figure 4.3.6; the corresponding dynamics of the queue sizes are shown in Figure 4.3.7.

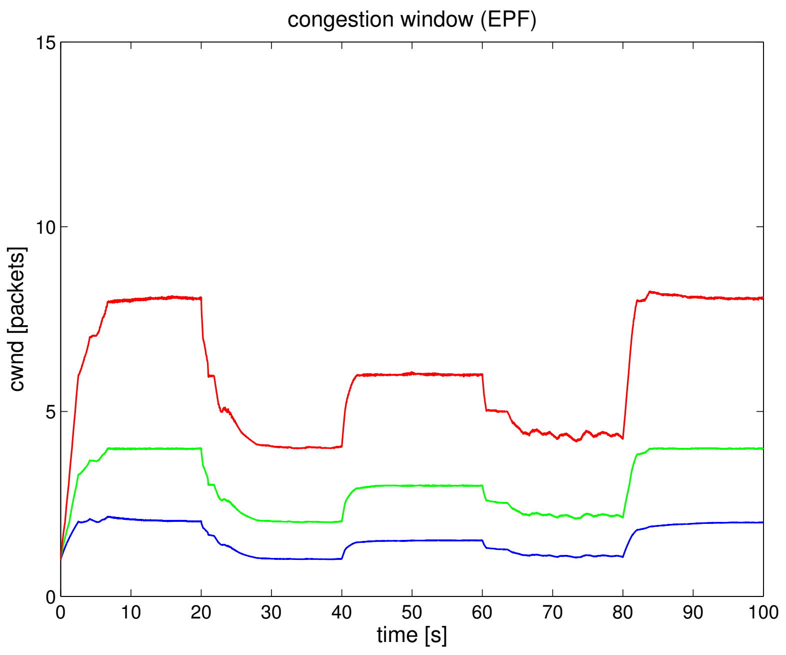
Since the load on the network is varied over time, the size of the congestion window should inversely follow the load as was theoretically shown in Figure 4.3.3. For CP-TCP/EPF displayed in Figure 4.3.6c, this is exactly the case. It is the only TCP variant that neatly adapts to the changes of the conditions. The other two variants, TCP NewReno with drop-tail queue and



(a) TCP NewReno/Droptail

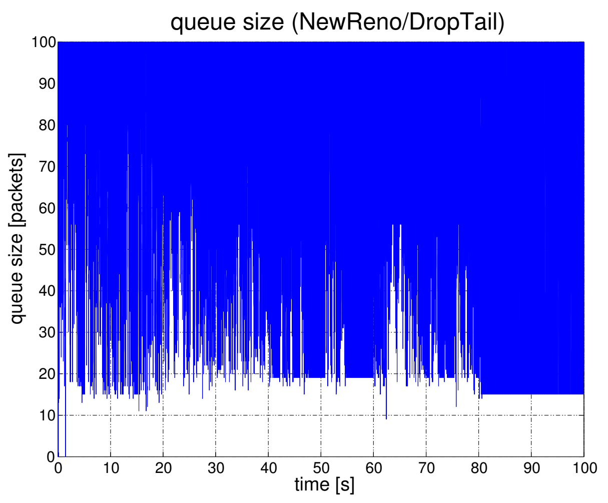


(b) TCP NewReno/RED

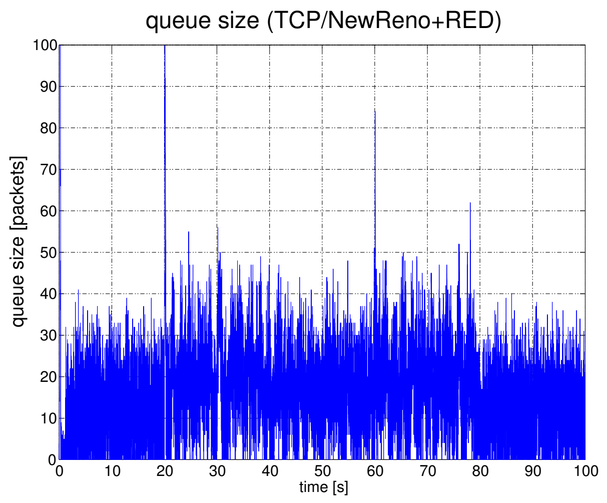


(c) CP-TCP/EPF

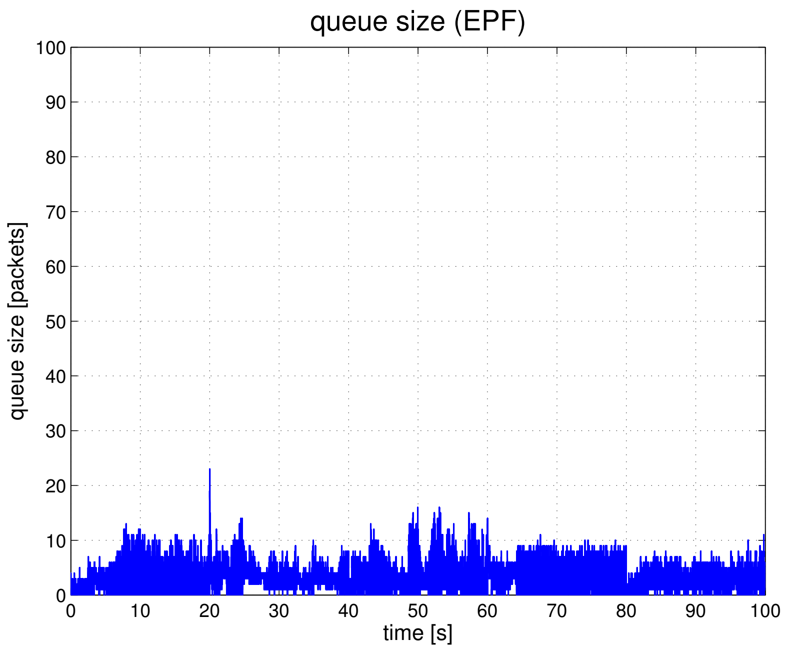
Figure 4.3.6: Congestion windows (flows 1, 5, 9) [Zim03]



(a) TCP NewReno/Droptail



(b) TCP NewReno/RED



(c) CP-TCP/EPF

Figure 4.3.7: Queue sizes at core link of router R1 [Zim03]

TCP NewReno with RED, also react to the load changes, but there are strong oscillations visible. Further looking at the queue size, CP-TCP/EPF is the only variant that is able to maintain low queue sizes over time also when the network conditions change. Even though the queue size remains much lower, the link utilization is still clearly above 90% and thus very good (cf. Table 4.3.3).

4.3.3 Conclusions

Two very important properties of CP-TCP/EPF were shown by the simulations. CP-TCP/EPF establishes a weighted proportionally fair rate allocation, and the sources adapt well to changing network conditions. While it is debatable what type of rate allocation is “fair” or “good”, from a point of traffic engineering it is important to be able to calculate resulting rate distributions. With CP-TCP, this is possible. Further in an actual network, conditions and demands change all the time. The congestion control algorithm should thus be able to adapt and work well over the full range of possible demands. Again, CP-TCP does this much better than conventional TCP variants.

4.4 Scalability and Fairness

4.4.1 Simulation Setup (Parking Lot Network)

In the previous section it was already shown that CP-TCP/EPF can adapt to changing network conditions. However, scalability of the algorithm with regard to the number of bottleneck links, delays, and network capacity is also important. This aspect shall be dealt with using a different simulation setup with a more realistic simulation scenario. The second simulation setup utilizes a multi-link “parking lot” topology (cf. Figure 4.4.1). Most published simulations so far were

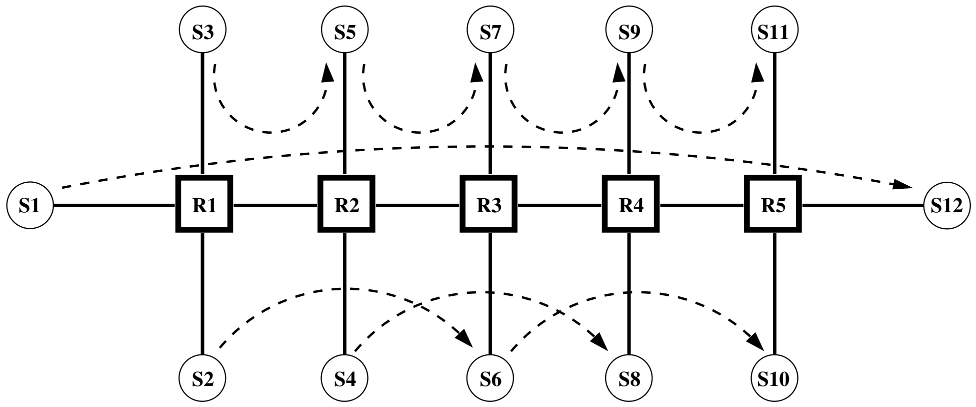


Figure 4.4.1: Parking lot network topology

conducted using only a small number of flows. For this reason a simulation scenario using a relatively large number of flows is additionally considered here. To evaluate scalability of Congestion Pricing based TCP with regard to the number of active flows and network capacity, the number of flows and the link capacities is scaled up without changing any of the other parameters. For good scalability, no further parameter changes should be necessary. Fairness between users is also evaluated.

In a “few flows” simulation scenario, up to 64 active flows are competing for bandwidth, while in a ”many flows” simulation scenario up to 6400 competing flows are used. The link capacities are dimensioned in such a way that only the core links are limiting. All core links have a capacity of 48 Mbps and a delay of 8 ms, except for the link between routers 2 and 3, which has a delay of 24 ms. The edge links have a capacity of 1000 Mbps and a delay of 1 ms. Forward and reverse paths are symmetric. These capacities allow a theoretical average of five packets in transit per connection.

For this simulation, a slightly changed source algorithm is used. The modified source algorithm limits the amount of congestion window change to 1 per acknowledgment (5.2.7). This is basically a limitation of the step size at which the steady-state is reached. The modification is done to make the results comparable to the Random Exponential Marking (REM) variant, which will be presented in detail in Section 5.2. The link algorithm parameters are $\gamma = 0.1$ and $\alpha = 0.002$[3], with the *willingness to pay* being 20 for all sources. All sources are greedy and use a packet size of a 1000 bytes.

Eight different flow paths with different hop counts and with different minimum round-trip delays are considered, as shown in Table 4.4.1. Each one of the eight sources (S1-S7,

Table 4.4.1: Connections and minimum round-trip time

Connection	S3→S5	S5→S7	S7→S9	S9→S11
hop count	2	2	2	2
min. RTT	20 ms	52 ms	20 ms	20 ms
Connection	S2→S6	S4→S8	S6→S10	S1→S12
hop count	3	3	3	5
min. RTT	68 ms	68 ms	36 ms	100 ms

S9) produces up to eight flows, yielding a maximum of 64 flows simultaneously active on the network.

For the “many flows” scenario, the number of flows is scaled up by a factor of 100. To accommodate the increased load, the capacities of the links are also scaled up by a factor of 100. Everything else remains unchanged. To examine the dynamic behavior, the number of active flows per source is again varied over time by turning on and off several flows every 20 seconds (cf. Figure 4.4.2 and Table 4.4.2). As explained before, optimally the congestion window should react inversely to the number of active flows (cf. Figure 4.3.3). The desired target queue size b_0 is chosen such that the corresponding queuing delay equals 3 ms. Thus for the “few flows” scenario: $b_0 = 18$ and for the “many flows” scenario: $b_0 = 1800$.

[3]Although a larger value for α was suggested in [AL00], the current queue size is weighted less here because oscillations were observed for larger α.

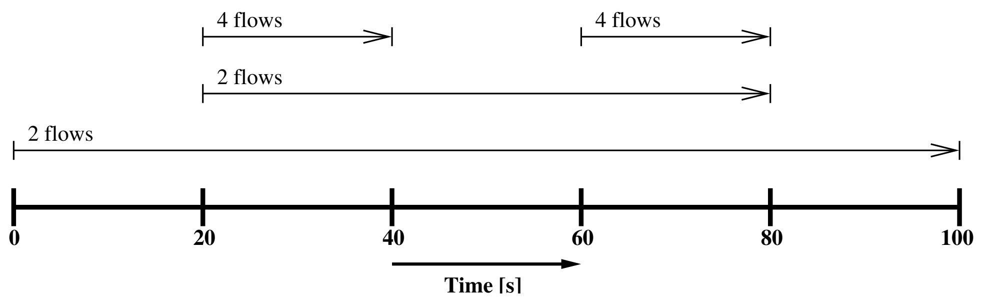


Figure 4.4.2: Start and stop times of the TCP flows per source

Table 4.4.2: Active flows at each edge node during the different time intervals

	0–20 s	20–40 s	40–60 s	60–80 s	80–100 s
Few flows scenario	2	8	4	8	2
Many flows scenario	200	800	400	800	200

4.4.2 Simulation Results

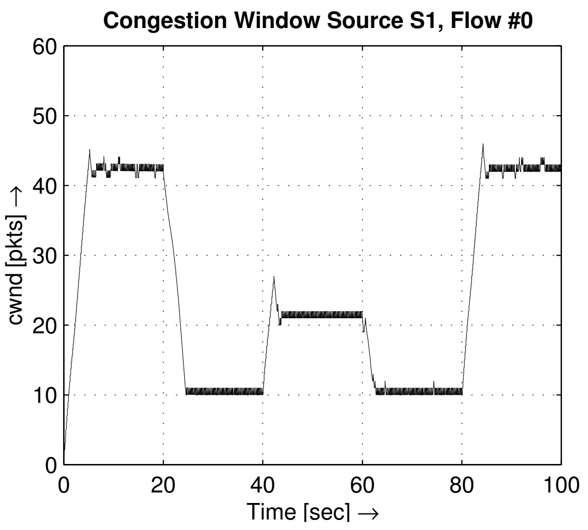
Figure 4.4.3a shows the variation of a congestion window over time for the "few flows" scenario. It follows nearly ideally the change of load as was seen before in the double bottleneck link topology (cf. Figure 4.4.3c). Figure 4.4.3b shows the trajectory of the instantaneous queue size at router R3 for the "few flows" case. Corresponding link performance measures are given in Table 4.4.3. Utilization is nearly perfect while the average queue size is only slightly larger

Table 4.4.3: Link performance measures for queue R3 ("few flows" scenario)

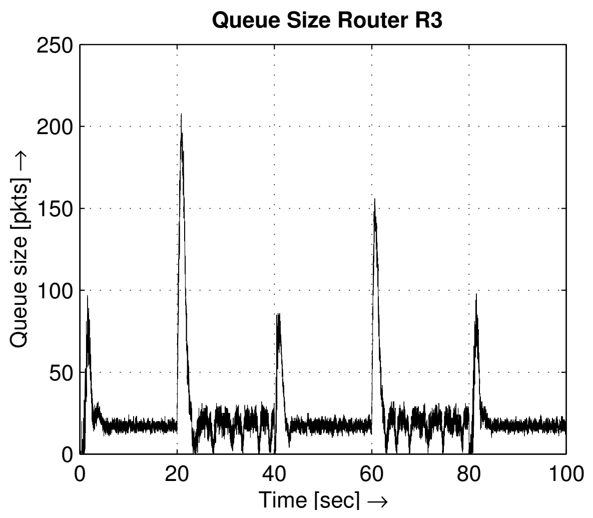
Time interval	0–20 s	20–40 s	40–60 s	60–80 s	80–100 s
Utilization	98 %	100 %	99 %	100 %	98 %
Avg. backlog	20 pkts	30 pkts	20 pkts	26 pkts	20 pkts

than the target queue size. Only when network conditions change and additional flows are turned on or off, the queue size increases for a short time until it returns to the desired equilibrium. Also note that in the time intervals 20–40 s and 60–80 s, there are oscillations visible at a low frequency. This is a sign of instability which will be examined more in detail for a different TCP variant in Chapter 6 using a control theoretic model. But even with these oscillations, performance is nearly perfect and by far better than with conventional TCP variants.

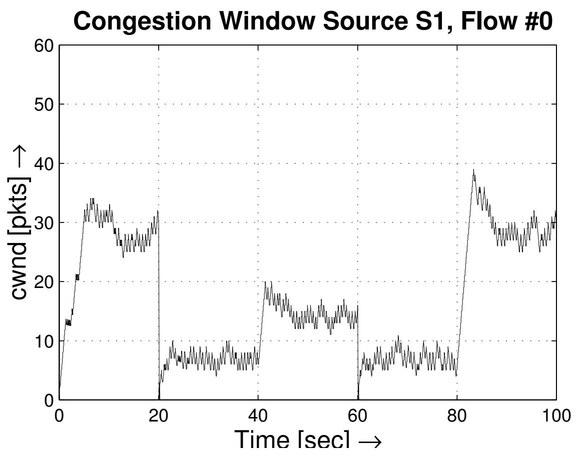
Figures 4.4.3c and 4.4.3d show the corresponding plots for the "many flows" scenario. The link performance parameters are shown in Table 4.4.4. With the scaled up number of flows and capacity, more oscillations are visible, but on average the behavior is still similar to the "few flows" case. The link utilization is well above 90% (cf. 4.4.4) and the average backlog is acceptable and even below the target. Thus, CP-TCP/EPF scales with regard to the number of flows over a wide range without adaptation of parameters.



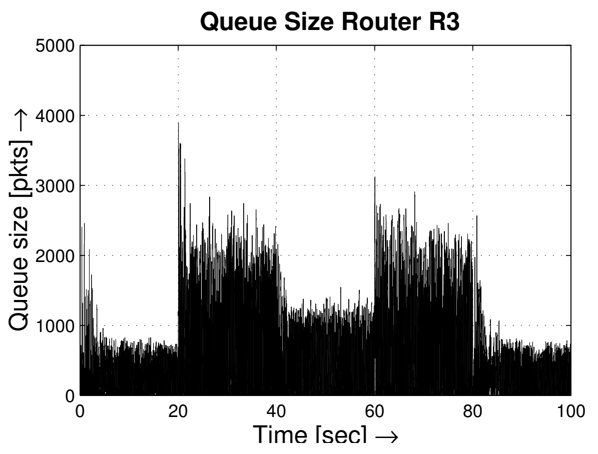
(a) Congestion window (few flows)



(b) Queue size (few flows)



(c) Congestion window (many flows)



(d) Queue size (many flows)

Figure 4.4.3: Congestion window and queue size (no slow start) [ZHK01]

Table 4.4.4: Link performance measures for queue R3 (“many flows” scenario) [ZHK01]

Time interval	0–20 s	20–40 s	40–60 s	60–80 s	80–100 s
Utilization	94 %	94 %	94 %	94 %	95 %
Avg. backlog	279 pkts	840 pkts	471 pkts	830 pkts	283 pkts

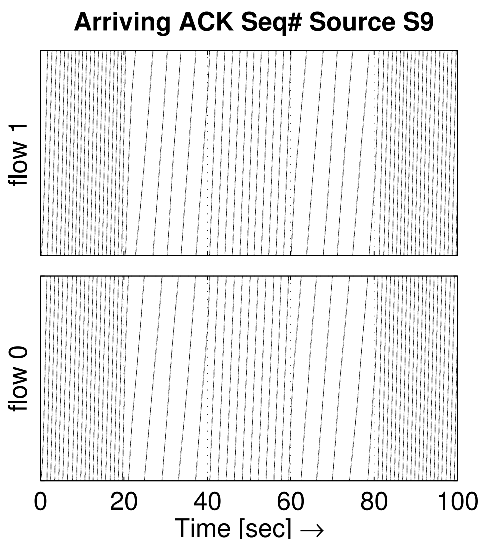
4.4.3 Fairness

In order to evaluate fairness between different users qualitatively, the numbers of the arriving acknowledgments at the sources were also recorded over time. Each arriving acknowledgment indicates the successful transmission of a segment with a size of 1000 bytes. Thus, for optimal fairness, the acknowledgment number should grow at the same speed for all flows that are using the same path. This is displayed in Figure 4.4.4: Time is shown on the x–axis, and for two exemplary flows the acknowledgment numbers are shown on the y–axis modulo 2000 for better readability. This notation results in parallel lines, where the slope is steeper for higher transmission rates. If the slope is the same for both flow 0 and flow 1 of the same source, both flows will receive the same rate. Rate distribution is then fair. When the load on the network changes, the transmission rate is adapted and thus the slope of the parallel lines changes too. Figure 4.4.4 also indicates that CP-TCP/EPF results in a perfectly fair rate distribution between flows of the same source even if network conditions change.

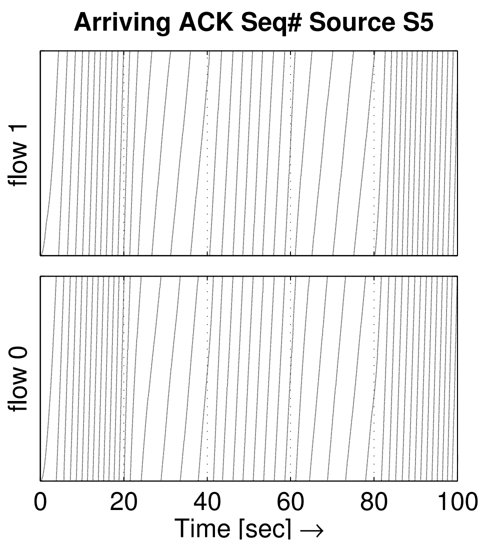
The results of the these experiments have been published more in detail in [ZHK01].

4.5 Conclusions

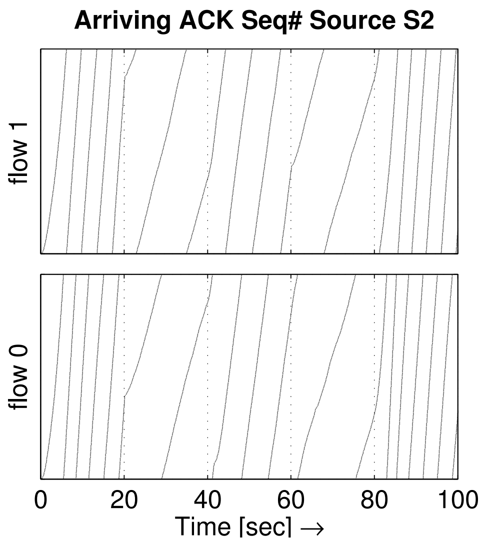
Congestion Pricing based TCP with Explicit Price Feedback (CP-TCP/EPF) is a practical implementation of Congestion Pricing theory for actual packet networks. It outperforms conventional TCP by far, even if modern Active Queue Management algorithms such as RED are used. CP-TCP/EPF can quickly adapt to changing network conditions and exhibits less oscillations. At the same time, the average queue size is much lower than with conventional TCP, while the link utilization is well above 90%. The rate allocation of CP-TCP/EPF is *weighted proportionally fair*, allowing the user to use different priorities for different applications. And in contrast to the conventional TCP variants, each user using the same parameters and network links receives the same rate, thus yielding fairness between users. CP-TCP/EPF is a very powerful congestion control implementation that does not only optimally use resources, but can also support traffic engineering by its calculable rate allocation.



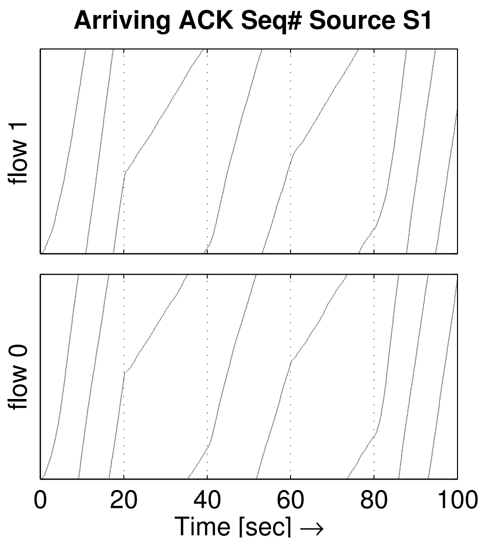
(a) Source 9



(b) Source 5



(c) Source 2



(d) Source 1

Figure 4.4.4: Acknowledgment sequence number plots (“many flows” scenario) [ZHK01]

Chapter 5

Single Bit Marking Strategies

As was shown in the previous chapter, a Congestion Pricing based TCP implementation with Explicit Price Feedback and Direct Window Updates (CP-TCP/EPF) performs much better than conventional TCP variants. However, since it requires significant changes to all existing layer 2 and 3 protocols on the Internet, it is highly impractical to deploy. For this reason, other implementation possibilities were proposed that require fewer modifications of the network and in the end hosts. The greatest problem is the encoding of the pricing information in the IP header. Without any changes to current ECN-enabled IP [RFB01], only one bit can be used to convey that information. The challenge is to create a TCP variant that uses Congestion Pricing and needs only a single bit to convey the pricing information. In this chapter, two proposals to solve this problem are presented. Their performance is then compared to CP-TCP/EPF. The first proposal, discussed in Section 5.1, is R. Gibbens' and F. Kelly's *Virtual Queue Mechanism (VQM),* which only allows 0 and 1 as the path price. Therefore, it does not allow transmission of the sum of the shadow prices. S. Athuraliya and S. Low, on the other hand, proposed a *Random Exponential Marking (REM)* strategy to convey the sum of the shadow prices using only a single bit. This second approach is discussed in Section 5.2. Finally, beginning with Section 5.4, the "Single Bit Resource Marking" approach developed by the author of this dissertation is presented.

5.1 Virtual Queue Mechanism (VQM)

5.1.1 VQM Algorithm

Path Price Transport

This implementation, referred to as "Virtual Queue Mechanism"[1], was initially proposed by R. Gibbens and F. Kelly [GK99]. It uses a single bit for marking, in other words the *path price* can only take two values: zero and one. For this reason, the *path price* is no longer a measure of the amount of rate change that is required, it can only signal the direction. The consequence is a violation of the additive path price property expressed in equation (3.2.15). For VQM, that

[1]R. Gibbens and F. Kelly presented in [GK99] a marking method that uses a "virtual" queue. However, they did not name it. Here the name VQM was chosen to refer to their method.

equation becomes:

$$p_n = \max_{l \in \mathcal{L}} A_{ln} \lambda_l(y_l), \tag{5.1.1}$$

where $\lambda_l \in \{0,1\}$. Thus, one can expect that VQM will "punish" connections that have more than one bottleneck link on their path.

Link Algorithm

R. Gibbens and F. Kelly proposed the use of a "virtual queue" for marking packets [GK99]. The packets are marked as follows: For every queue in a network node, a second "virtual queue" is run with a scaled down capacity and a lower service rate but identical packet input. If this virtual queue exceeds its capacity, all packets in the real queue are marked until the virtual queue becomes empty. The scaling factor is defined as follows:

$$\begin{aligned} f_{scale} &= \frac{virtual\,service\,rate}{actual\,service\,rate} \\ &= \frac{virtual\,queue\,capacity}{actual\,queue\,capacity}. \end{aligned}$$

The maximum queue size q_{max} defines the actual capacity of the queue.

The Virtual Queue Mechanism was derived from a cost function $C(y_l)$ that represents the rate of loss.

Source Algorithm

VQM was initially proposed for synchronous networks where sources can directly set their rates. The original source algorithm is:

$$x_n(t+1) := x_n(t) + \kappa(w_n - p_n(t)x_n(t)),$$

where $p_n = 1$ if the packet was marked, 0 otherwise. For a TCP/IP based packet network, the source algorithm must be changed. To make VQM applicable for packet based, asynchronous networks, the source algorithm must be modified accordingly. This was already done in Subsection 4.2.1 for CP-TCP/EPF. There, equation (4.2.3) was derived:

$$\Delta cwnd_n(t) \cong \bar{\kappa}_n \left(w_n \cdot \frac{RTT_n(t)}{cwnd_n(t)} - p_n(t) \right).$$

In this case, however, the *path price* p_n can only take the values 0 and 1. For this reason, following a suggestion by P. Key [Key01], the window update algorithm was slightly modified and implemented it as follows:

For every received non-marked acknowledgment, the congestion window is increased by:

$$\Delta cwnd = \overline{\kappa} \cdot \frac{RTT}{cwnd} \cdot w_n \tag{5.1.2}$$

and for every marked acknowledgment, differing from (4.2.3), the congestion window is decreased by:

$$\Delta cwnd = \overline{\kappa}. \tag{5.1.3}$$

This way of implementing the source algorithm more closely resembles current TCP variants that use the *Additive Increase Multiplicative Decrease (AIMD)* property (cf. Subsection 2.3.3). This property describes the development of the congestion window for every full round, thus for every full window of acknowledgments and under the assumption of roughly constant round-trip time (RTT).

In actual TCP implementations, in *congestion avoidance,* the congestion window is only increased after a full window of acknowledgments has been received, thus *cwnd* remains constant for one full round-trip time (RTT). If the same is applied here, for every full congestion window of unmarked acknowledgments, the congestion window update becomes:

$$cwnd_{new} = \quad cwnd_{old} + \overline{\kappa} \cdot RTT \cdot w_n \quad \text{(full window of unmarked acknowledgments)},$$

which clearly shows the *additive increase* property. For a full window of marked acknowledgments, on the other hand, from (5.1.3) and under the condition $0 < \overline{\kappa} < 1$ the window becomes:

$$cwnd_{new} = \quad (1 - \overline{\kappa}) \cdot cwnd_{old} \quad \text{(full window of marked acknowledgments)},$$

which shows the *multiplicative decrease* property. For example, for $\overline{\kappa} = 0.5$ the congestion windows is halved. Again, this is similar to current TCP variants using the ECN addition, that halve the window for the first marked acknowledgment and then ignore marks for a full round-trip time [RFB01].

This implementation was also chosen for another reason: Note that p_n can be 1 at most. In order for the original source algorithm (4.2.3) to be applicable, the following requirement must be met:

$$w_n \cdot \frac{RTT^*}{cwnd^*} \overset{!}{<} 1.$$

Thus, one must be careful not to choose too large *willingness to pay* parameters. This is not a requirement for the alternative source algorithm (5.1.2)–(5.1.3). It is therefore more robust against bad parameter choices.

However, there is a side-effect to these advantages: The modified implementation is based on a slightly changed utility function. In equilibrium,

$$U'(x_n^*) = p_n^*, \tag{5.1.4}$$

as can be derived from the derivative of (3.2.8)–(3.2.9). With the original source algorithm (4.2.3), in equilibrium $\Delta cwnd = 0$, and thus in equilibrium:

$$w_n = p_n^* x_n^*. \tag{5.1.5}$$

Inserting (5.1.5) into (5.1.4) yields:

$$U'(x_n) = \frac{w_n}{x_n},$$

which finally leads to the desired logarithmic utility function

$$U(x_n) = w_n \ln(x_n).$$

However, when using the modified source algorithm (5.1.2)–(5.1.3), which can be rewritten as:

$$\Delta cwnd = \overline{\kappa} \cdot \frac{RTT(t)}{cwnd(t)} \left((1 - p_n(t))\, w_n - p_n(t) \cdot \frac{cwnd(t)}{RTT(t)} \right),$$

in equilibrium

$$w_n - p_n^* w_n = p_n^* \cdot x_n^*,$$

and thus

$$U'(x_n) = \frac{w_n}{x_n + w_n},$$

which leads to a slightly different utility function

$$U(x_n) = w_n \ln (x_n + w_n).$$

This also has impact on the resulting rate allocation. Using the same double bottleneck link scenario as in Subsection 3.2.6, and using the modified utility functions with equal *willingness to pay* w, the rate allocation is established solving (3.2.1)–(3.2.3):

$$\begin{aligned}
\text{maximize} \quad & \sum_{n=1,2,3} U(x_n^*) = 2w \ln(\alpha C + w) + w \ln((1-\alpha)C + w) \\
\text{under the constraint} \quad & \alpha < 1 \\
\text{over} \quad & \alpha > 0 \\
\Longrightarrow \quad & \frac{2wC}{\alpha C + w} - \frac{wC}{(1-\alpha)C + w} \overset{!}{=} 0 \\
\Longrightarrow \quad & \alpha = \frac{2 + \frac{w}{c}}{3}.
\end{aligned}$$

For this reason, the modified implementation only leads to proportionally fair rate allocations for $w \ll c$. Nonetheless, proportionally fair rate allocation is a reasonable assumption as commonly, capacity c is very large and w should be chosen such that

$$\sum_n w_n \overset{!}{<} c_{bottleneck}.$$

For realistic network scenarios and in the simulations conducted, this is the case.

5.1.2 Performance of VQM (Double Bottleneck Link Network)

For performance evaluation, the same double bottleneck link simulation setup that was presented in Subsection 4.3.1 is used. As before, two different scenarios are investigated to evaluate the resulting rate allocation and the dynamic behavior. For the VQM simulations, the following parameters were chosen: $f_{scale} = 0.9$, $q_{max} = 300$.

Rate Allocation

The established rate allocation is shown in Figure 5.1.1. Again, the horizontal lines indicate the theoretical values of a *weighted proportionally fair* rate allocation. Flows on path III, the one

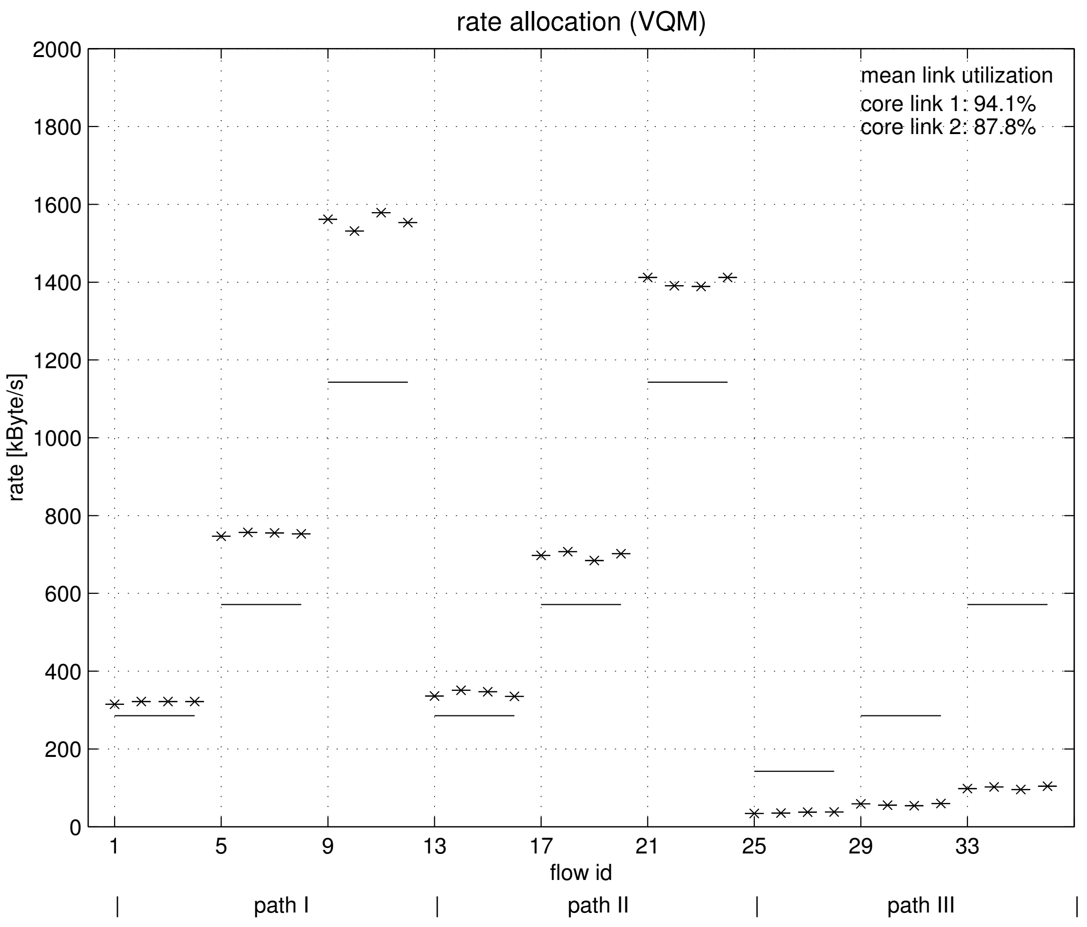


Figure 5.1.1: Rate allocation using VQM [ZK02]

that uses two bottleneck links, receive a significantly lower rate than the theoretical *weighted proportionally fair* rate allocation. This is an expected result of the changed path price property (5.1.1). The remaining bandwidth is then distributed to the paths that use only one bottleneck link. On the other hand, VQM is still able to establish a service differentiation using different *willingness to pay* parameters. Also, there is only slight dependency on round-trip time as the similar rate allocations on paths I and II indicate. Thus, VQM has some significant advantages over conventional TCP variants at comparable bottleneck link utilization (cf. Table 5.1.1).

Table 5.1.1: Bottleneck link utilization

TCP variant	Utilization of core link 1	Utilization of core link 2	Average core link utilization
TCP NewReno+drop-tail	96.5 %	87.5 %	92.0 %
TCP NewReno+RED	95.1 %	89.9 %	92.5 %
CP-TCP/EPF	93.4 %	92.6 %	93.0 %
VQM	94.1 %	87.8%	90.9%

Because the connections on path III receive a much lower bandwidth share than expected, there is a risk that these connections will suffer from starvation. Even a larger *willingness to pay* does not prevent starvation (cf. Figure 5.1.1, flows 33–36). If more bottleneck links are used, this risk will become even worse. This is a significant disadvantage, as for this reason VQM does not scale to larger networks with many (potentially congested) links. Therefore the parking lot topology is omitted here. VQM was evaluated in a parking lot network in [Ham01]. Some representative results are summarized in Section 5.4.4.

Dynamics

To evaluate the dynamic behavior, the number of active flows over time was again changed as was described in Subsection 4.3.1. The resulting congestion windows for flows 1, 5, and 9 are shown in Figure 5.1.2. Since different *willingness to pay* parameters are used for the three flows, the congestion windows must be of different size. Just like CP-TCP/EPF (cf. Figure 4.3.6c),

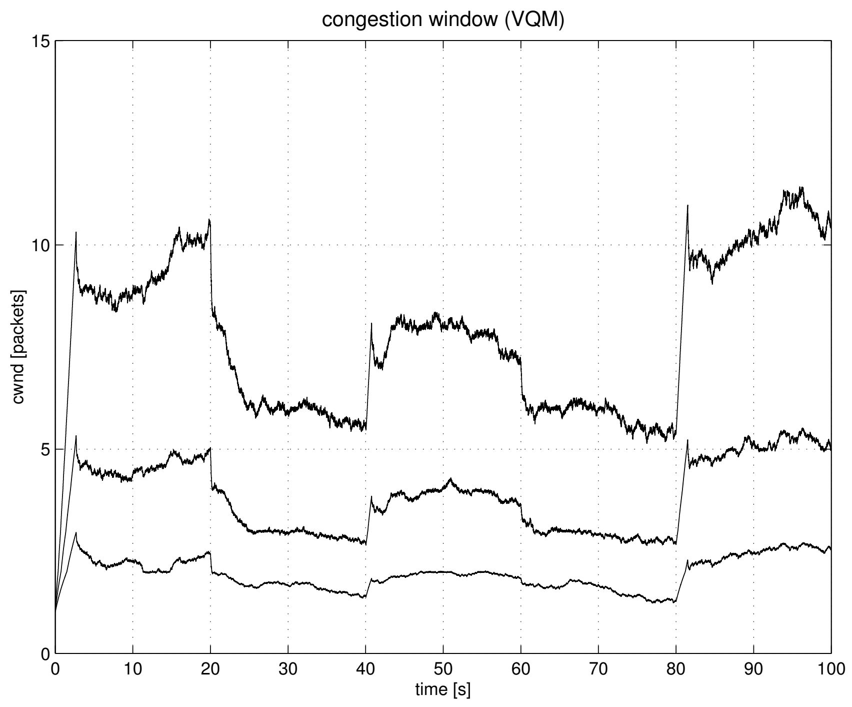


Figure 5.1.2: Congestion window (flows 1, 5, 9) [ZK02]

the changes of the window size are smooth. The exact behavior depends on the parameter $\overline{\kappa}$. Note, however, that VQM allows changes of the congestion window that are only a fraction of the packet. Such a change will not change the sending rate because TCP will only be able to send an additional packet if a full packet (of maximum segment size) fits into the congestion window.

Figure 5.1.3 shows the size of the queue of the core link at router R1. Similarly to CP-TCP/EPF (cf. Figure 4.3.7c), the queue size is very low even though utilization is around 90% (cf. Table 5.1.1). Peaks are visible only at times when additional flows are turned on. Because of the delay in the control loop, such peaks cannot be avoided.

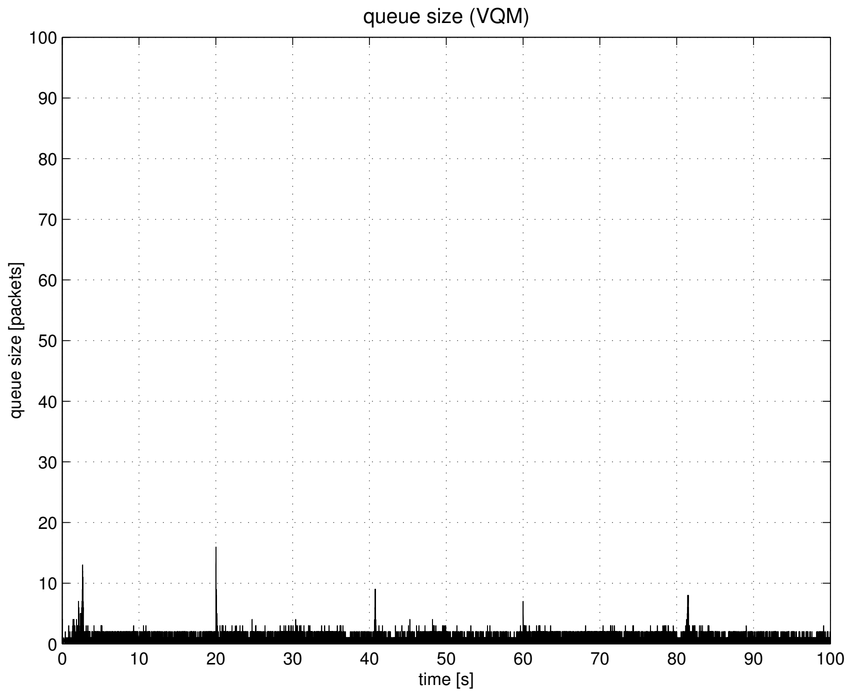


Figure 5.1.3: Queue sizes at core link of router R1

5.2 Random Exponential Marking (REM)

5.2.1 REM Algorithm

Path Price Transport

Random Exponential Marking (REM) was suggested by S. Athuraliya and S. Low [AL00, ALLY01]. It uses a single bit in the IP header to encode the shadow price. The key idea of REM is the use of an *exponential* marking function. The shadow price $\lambda_l(t)$ is encoded using the marking probability $m_l(t)$ at time t:

$$m_l(t) = 1 - \phi^{-\lambda_l(t)}. \tag{5.2.1}$$

An exponential function was suggested because it has the desirable property that the complement function of the end–to–end marking probability

$$m_n(t) = 1 - \prod_{l \in \mathcal{L}(n)} (1 - m_l(t))$$

yields the aggregate path price:

$$p_n(t) = -\log_\phi(1 - m_n(t)) = \sum_{l \in \mathcal{L}} A_{ln}\lambda_l. \tag{5.2.2}$$

Thus, by estimating the marking probability $\widehat{m}_n(t)$, the source can calculate the path price $\widehat{p}_n(t)$, which is an estimate of the sum of the shadow prices λ_l on the path as required by (3.2.15). As opposed to the Virtual Queue Mechanism (VQM), the path price can still take any value, although only one bit is used to convey that information. However, for this to work, all network components must use an identical marking function $m_l(\lambda_l)$. This implies that they all must agree on the same ϕ. Additionally, as the authors have shown, the performance is best if $0.2 < m_n < 0.97$.

In order to estimate the end–to–end marking probability $\widehat{m}_n(t)$ required for the calculation of the path price $\widehat{p}_n$, a history of several packets is needed.

Link Algorithm

REM's link algorithm, the *shadow price computation* rule PC3 (4.2.4), was already introduced in Subsection 4.2.2. According to that rule, each link's shadow price λ_l is increased or decreased using two components: the current difference in input rate and capacity of the link, and the instantaneous queue size. The other two proposed price computation rules PC1 and PC2 only use one of these components [AL00]:

$$\text{PC1:} \quad \lambda_l(t+1) = [\lambda_l(t) + \gamma(\widehat{y}_l(t) - c_l)]^+ \tag{5.2.3}$$

$$\text{PC2:} \quad \lambda_l(t) = [\gamma(b_l(t) - b_0)]^+ \tag{5.2.4}$$

$$\text{PC3:} \quad \lambda_l(t+1) = [\lambda_l(t) + \gamma(\alpha_l(b_l(t) - b_0) + \widehat{y}_l(t) - c_l)]^+ . \tag{5.2.5}$$

PC1 only uses the excess input bandwidth to compute the current shadow price. Thus, the congestion measure is completely decoupled from performance measures such as queue length. Such coupling of congestion and performance measures is not desired, as was described in Subsection 2.4.1. However, PC1 in equilibrium only ensures that input and output rates match. It cannot ensure that this equilibrium happens at zero queue size. PC2 can be derived from PC1 as the queue size is an integrated measure of excess input bandwidth. Here, in equilibrium the queue size will be b_0. PC2, however, couples congestion measure and performance measure. PC3 finally combines both approaches. Since PC3 is the recommended algorithm by the original authors, it will be used for the purpose of performance evaluation of REM. Later, in Subsection 5.4.4, results from simulations including all three variants will be presented.

Source Algorithm

S. Athuraliya and S. Low [AL00, ALLY01] initially proposed to derive the REM source algorithm from the TCP Vegas source algorithm. TCP Vegas, in congestion avoidance, either decreases, stays constant, or increases the congestion window by one segment, depending on some threshold for the difference of current and minimum round-trip time [BP95]. In [LPW01] the idea was presented to use the *willingness to pay* w_n, instead, as a threshold for the current charge $\widehat{p}_n(t)\widehat{x}_n(t)$. This is motivated by the first order condition of the social optimum (3.2.7) and the use of a logarithmic utility function (3.2.21): In equilibrium

$$w_n \overset{!}{=} p_n^* x_n^* . \tag{5.2.6}$$

Thus, in order to make the congestion window converge such that (5.2.6) is fulfilled, from (2.3.1) the update rule (run once per round-trip time) becomes:

$$cwnd(t+RTT) := \begin{cases} cwnd(t)+1 & \text{if } \hat{p}_n(t)\widehat{x}_n(t) < w_n \\ cwnd(t)-1 & \text{if } \hat{p}_s(t)\widehat{x}_n(t) > w_n \\ cwnd(t) & \text{if } \hat{p}_n(t)\widehat{x}_n(t) = w_n \end{cases} \tag{5.2.7}$$

where w_n is the *willingness to pay*, $\widehat{x}_n$ is the estimated sending rate for source $n \in \mathcal{N}$, and $\widehat{p}_n$ is the estimated path price for source n. This source algorithm will, in the following sections, be referred to as "*Vegas/CP*".

5.2.2 Performance of REM (Double Bottleneck Link Network)

Again, to evaluate REM/PC3's ability to establish the correct rate allocation and to cope with changing network conditions, simulations are performed in a double bottleneck link network (cf. Subsection 4.3.1). The TCP Vegas implementation in the *ns-2* network simulator was modified to use the new *Vegas/CP* window update rule (5.2.7). TCP Vegas also uses a special *Slow Start* algorithm [BP95], which was retained. For both simulations, the following parameter settings were chosen: $\phi = 1.06$, $\gamma = 0.001$, $\alpha = 0.1$, $b_0 = 2$. A history of 50 packets was used to estimate the end–to–end marking probability. Because of the different source algorithm, the *willingness to pay* parameters were chosen differently from the values shown in Table 4.3.2. Here, 1, 2, and 4 are used respectively.

Rate Allocation

Because REM uses random marking, the rate allocation simulation was repeated 30 times using different seeds for the random number generator. In Figure 5.2.1, the resulting mean rates and 95% confidence intervals are shown. Like before, the horizontal lines in the plots show the theoretical values for a *weighted proportionally fair* rate allocation. As can be seen from the results, REM can establish the theoretical rate allocation and separate the service classes according to the *willingness to pay* extremely well. It also achieves the highest bottleneck link utilization (cf. Table 5.2.1). Thus, at the steady-state, REM is a very good replacement for

Table 5.2.1: Bottleneck link utilization

TCP variant	Utilization of core link 1	Utilization of core link 2	Average core link utilization
TCP NewReno+drop-tail	96.5 %	87.5 %	92.0 %
TCP NewReno+RED	95.1 %	89.9 %	92.5 %
CP-TCP/EPF	93.4 %	92.6 %	93.0 %
VQM	94.1 %	87.8%	90.9%
REM	94.9%	93.2%	94.1%

CP-TCP/EPF.

However, the first four flows receive slightly higher rates than they should. This is caused by the source algorithm: Since REM can only increase and decrease the congestion window by

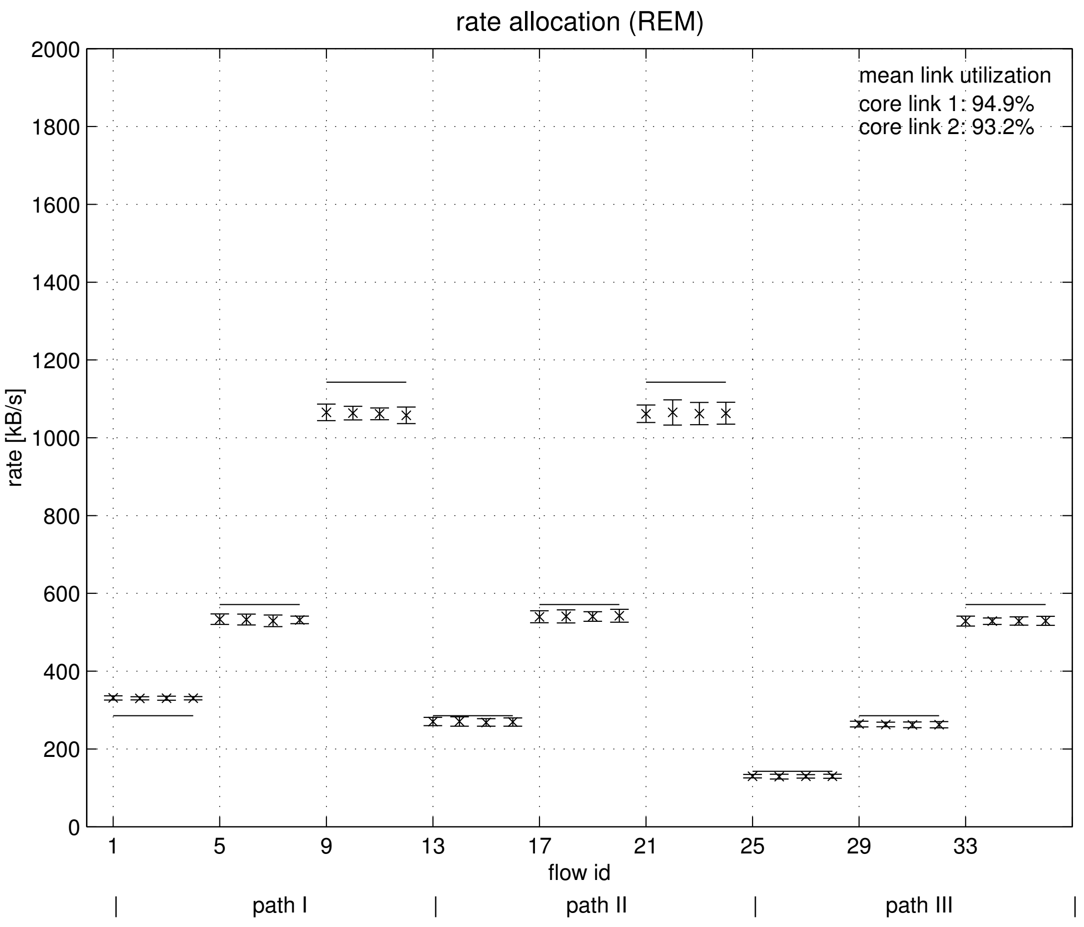


Figure 5.2.1: Rate allocation of REM with PC3 [ZK02]

one segment per round-trip time, it cannot establish the correct rates for the first four flows. The congestion window oscillates between one and two, causing the flows to receive more than their fair share of bandwidth. This effect becomes more obvious when the capacity of the core links is reduced. However, this problem is a result of the window based rate control, not necessarily of REM. Also, it can be observed that the rate allocation does not depend on the round-trip time; for both paths I (6 ms) and II (34 ms) the rate allocation is approximately identical. Thus, REM does establish a nearly *weighted proportionally fair* rate allocation.

Dynamics

Figure 5.2.2 shows the resulting congestion windows for flows 1, 5, and 9 when varying the number of active flows over time. Again, since different *willingness to pay* parameters are used for the three flows, the congestion windows must be of different size. This is the case even though the congestion window sizes are subject to oscillations. The oscillations are partly caused by the source algorithm because it adjusts the congestion window size by one full packet almost every round-trip time. Minor modifications to the source algorithm could lead to a more stable window. However, the oscillations are also caused by instability. The general impact of instability will be discussed more in detail in Chapter 6.

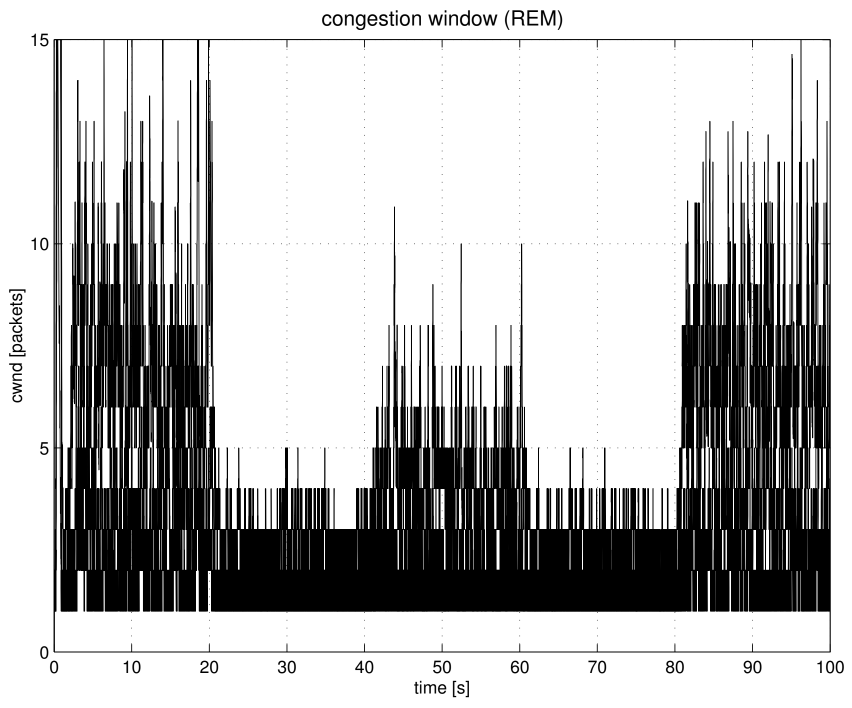


Figure 5.2.2: Congestion window (flows 1, 5, 9)

These oscillations are also visible in the queue trajectory of the core link at router R1, which is shown in Figure 5.2.3. Obviously, this is not desired as it introduces jitter and high peaks in the instantaneous queue size. On the other hand, higher queue sizes also lead to better bottleneck link utilization (cf. Table 5.2.1).

Thus, although REM can establish average throughput very well, it has weaknesses in its dynamic behavior. In particular, oscillations cannot completely be avoided. Compared to CP-TCP/EPF, this is a significant disadvantage.

5.2.3 Evaluation of REM's Path Price Estimation, Scalability and Fairness (Parking Lot Network)

Ways to Improve REM

As was shown, the reduction of pricing information to a single bit caused significant disadvantages in dynamic behavior when compared to the explicit price feedback used before. While VQM only allows a single bit of information, REM encodes the full path price in a special way. However, even REM has some problems in the dynamic behavior, as was shown. To improve REM, the encoding mechanism and transport algorithm of the path price were studied. Their impact on performance were evaluated in [ZHK01]. Here the significant findings and conclusions are summarized.

Marking Probabilities and Path Price Estimation

Because REM has to estimate the marking probability in order to recover the path price, a history of packets is needed. This introduces additional delay before the source can react to

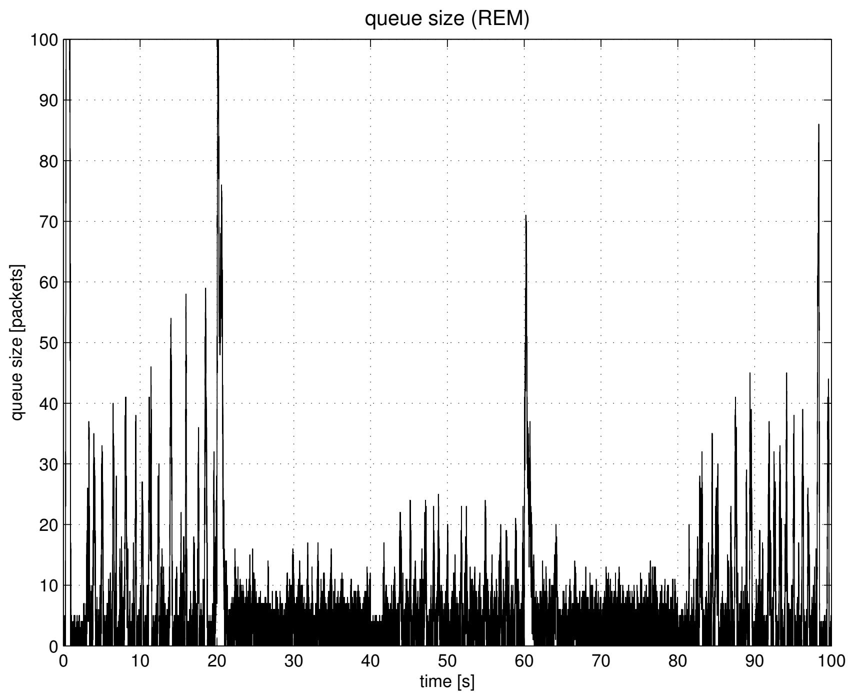


Figure 5.2.3: Queue size at core link of router R1

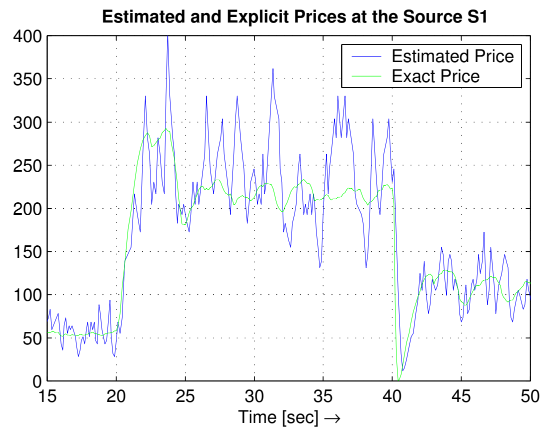
changed path prices. Additionally, if the path price estimation is wrong, this will result in incorrect rate allocation. Therefore, using the "parking lot" topology and flow scenarios that were introduced above in Subsection 4.4.1, the path price estimation was examined by means of simulation. The results gained from the simulations to evaluate scalability with regard to the number of active sources and fairness were used and compared to the results derived with the CP-TCP/EPF in Subsection 4.4.1. As a compromise between full pricing information and single bit marking, a multi-bit variant was also suggested and implemented in [ZHK01].

The performance of REM significantly depends on the end–to–end marking probability estimation. For very low or very high marking probabilities, the algorithm performs suboptimally and high oscillations can be observed [AL00]. While in small networks the parameter ϕ can be adjusted to get optimal marking probability, in large networks with highly fluctuating numbers of sources this will be a very difficult task. To overcome this problem, one could think of an adaptive version where the parameter ϕ is adjusted dynamically. However, this would require that all network components and sources agree upon the same $\phi(t)$ for all $t \geq 0$. Such a solution can hardly be implemented. Alternatively, a multi-bit variant could be used, where each bit is encoded using a different ϕ. Using four different values for ϕ, the shadow price at each link is translated into four different marking probabilities. The source then estimates the end–to–end marking probability for each bit and only selects the bit with an estimated probability in the "good range" where the algorithm performs well. To keep the algorithm as simple as possible, the bit with the marking probability closest to 0.5 was chosen for the calculation of the estimated path price.

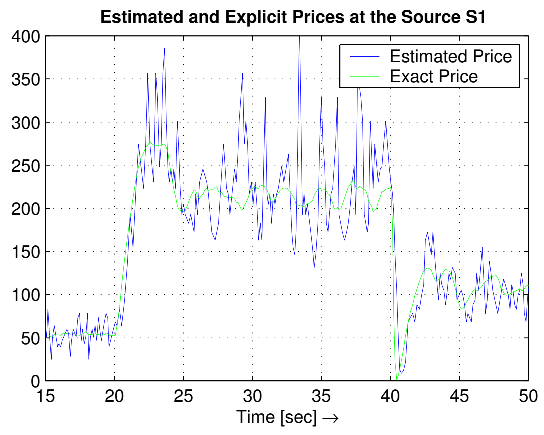
The different variants of REM were then simulated in a parking lot network in order to examine different marking strategies and to compare S. Low's original proposal for a single bit marking scheme with an extended version using four bits as described before. For these simulations, a history of 50 packets was used to estimate the end–to–end marking probability. The following parameters were used: for the single bit case $\phi = 1.007$, for the four bit case $\phi_1 = 1.06$, $\phi_2 = 1.007$, $\phi_3 = 1.001$, $\phi_4 = 1.00015$.

Simulation Results

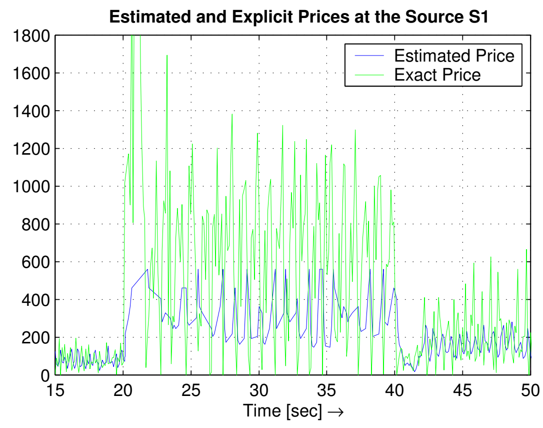
Figure 5.2.4 shows the estimated and actual path prices at source 1 over time for the standard single bit REM and the proposed four bit variant. This simulation confirms the problem with



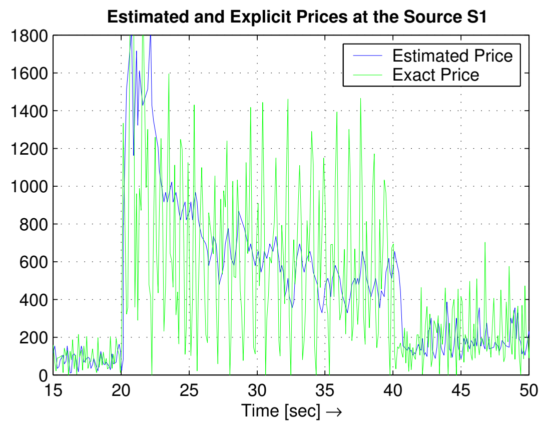
(a) 1 bit marking (few flows)



(b) 4 bit marking (few flows)



(c) 1 bit marking (many flows)

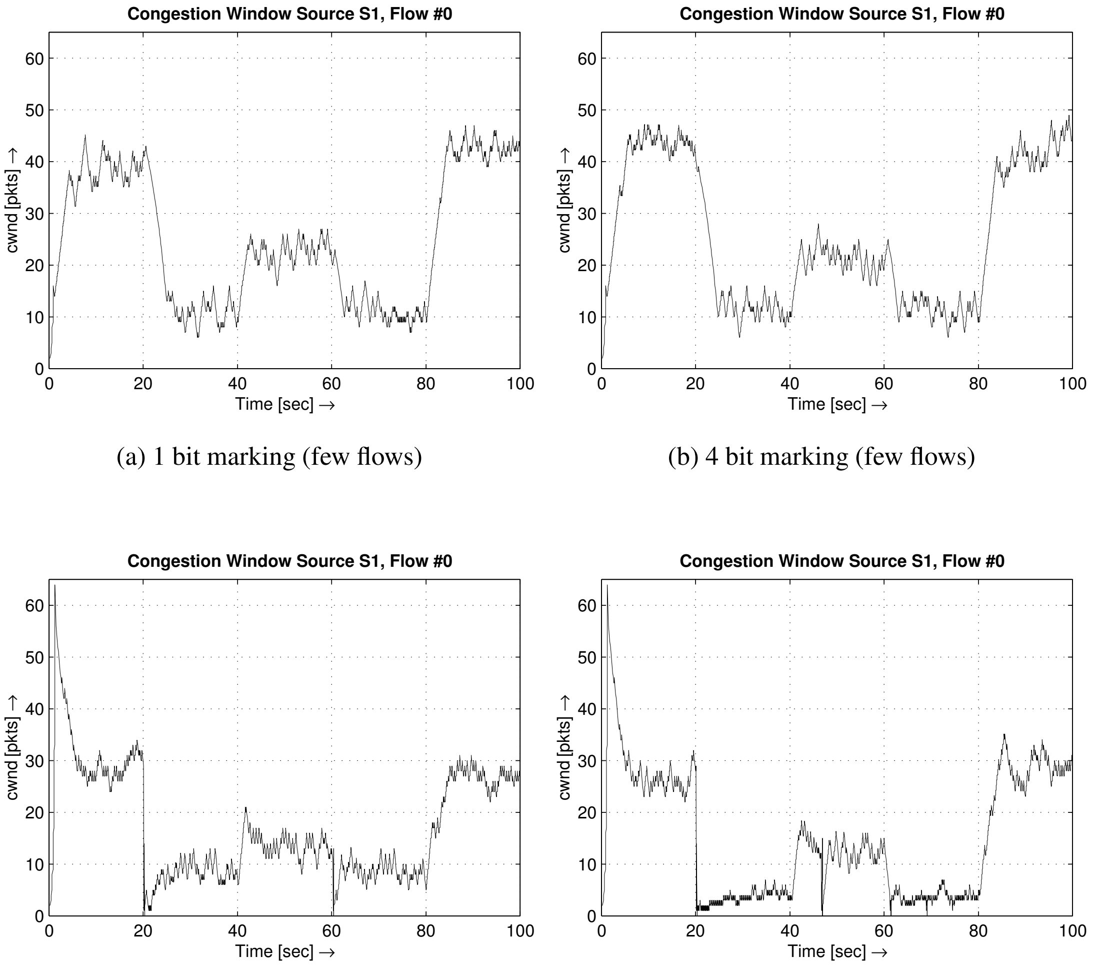


(d) 4 bit marking (many flows)

Figure 5.2.4: Price tracking [ZHK01]

the estimation of the path marking probability: Since a history of 50 packets is used to estimate the marking probability, an additional delay is introduced before the source can react to changed path prices. This is a problem especially if a source has only a small number of packets in transit. Thus, rate updates could be delayed several round-trip times. If the number of competing flows is increased, larger path prices are observed. If ϕ is not adjusted, the single bit variant cannot follow into these higher price regions. While the four bit variant cannot solve the delay introduced by the marking probability estimation, it allows the path price to be estimated correctly even if the number of flows is greatly increased (cf. Figure 5.2.4d). Thus, better performance of the 4 bit variant could be expected.

Congestion windows and queue lengths are shown in Figures 5.2.5 and 5.2.6. For both



(a) 1 bit marking (few flows)

(b) 4 bit marking (few flows)

(c) 1 bit marking (many flows)

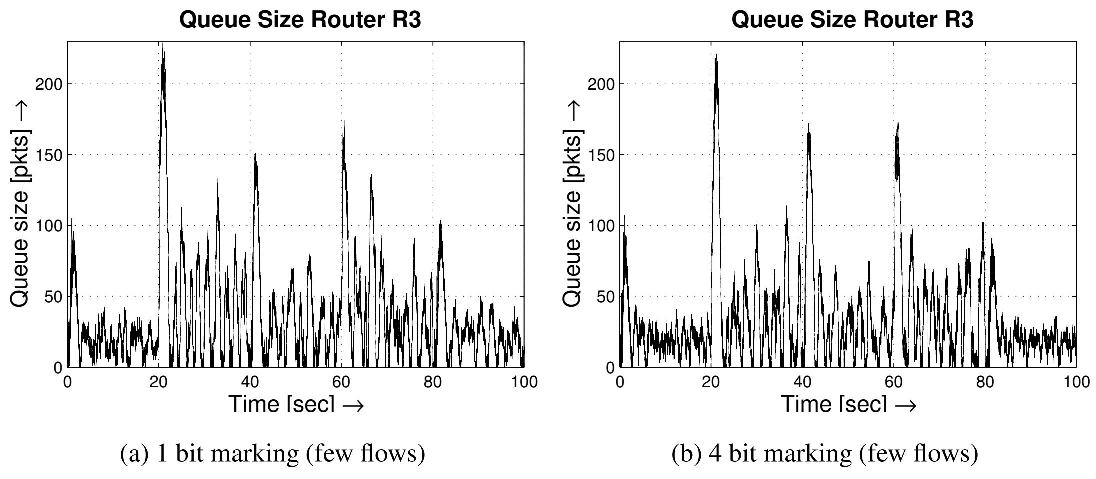
(d) 4 bit marking (many flows)

Figure 5.2.5: Congestion window size [ZHK01]

marking strategies and for both the “few flows” and the “many flows” scenario, the congestion window inversely follows the change of load. Also note that the oscillations are reduced in comparison to Figure 5.2.2. This is a result of the greater overall congestion window size where oscillations of one are not as visible. Further, an optimized parameter set was used.

Tables 5.2.2 and 5.2.3 show the link performance parameters for both marking strategies. The four bit marking scheme can on average establish a slightly higher throughput, and its average backlogs are closer to the target value. However, the differences are only marginal. Even though the single bit variant cannot track congestion prices as well as the four bit variant, performance values are similar. Considering only these parameters, the single bit variant seems sufficient even if network conditions change significantly.

In the Subsection 4.4.1, the performance of CP-TCP with Explicit Price Feedback was evaluated in the same network topology. Since full pricing information was used in this case, better



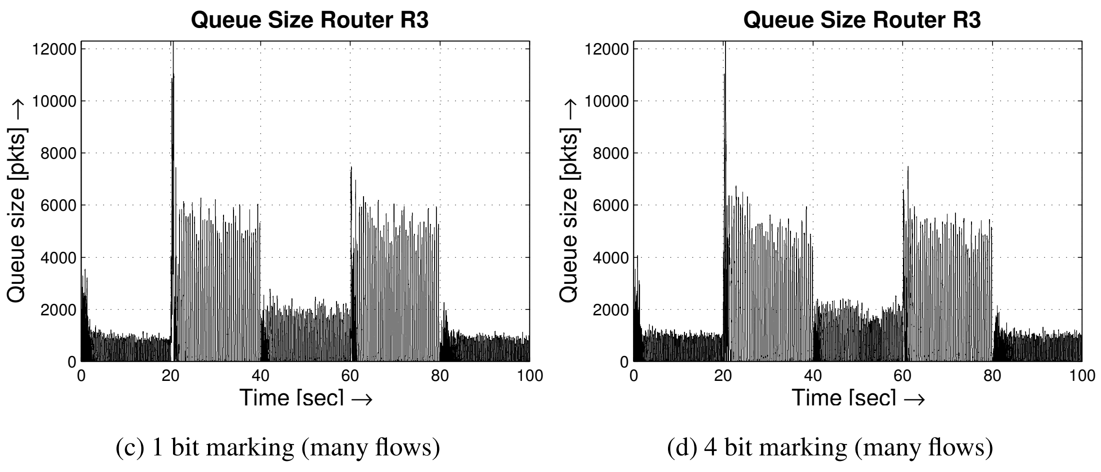


Figure 5.2.6: Queue size [ZHK01]

Table 5.2.2: Link performance measures for queue R3 (“few flows” scenario) [ZHK01]

Time interval	0–20 s	20–40 s	40–60 s	60–80 s	80–100 s
Utilization (1 bit)	99 %	96 %	97 %	97 %	96 %
Utilization (4 bit)	99 %	98 %	97 %	98 %	97 %
Avg. backlog (1 bit)	21 pkts	51 pkts	33 pkts	43 pkts	27 pkts
Avg. backlog (4 bit)	22 pkts	44 pkts	35 pkts	42 pkts	20 pkts

Table 5.2.3: Link performance measures for queue R3 (“many flows” scenario) [ZHK01]

Time interval	0–20 s	20–40 s	40–60 s	60–80 s	80–100 s
Utilization (1 bit)	95 %	86 %	90 %	86 %	94 %
Utilization (4 bit)	96 %	87 %	95 %	87 %	95 %
Avg. backlog (1 bit)	405 pkts	1738 pkts	638 pkts	1568 pkts	359 pkts
Avg. backlog (4 bit)	458 pkts	1769 pkts	739 pkts	1655 pkts	419 pkts

performance could be achieved. In the “few flows” scenario (cf. Table 4.4.3 for CP-TCP/EPF and Table 5.2.2 for REM), however, the link utilization is almost the same for both CP-TCP/EPF and REM. The average backlog, on the other hand, is on average 45% larger than in the case of CP-TCP/EPF. Especially in the case of many active flows REM performs worse. The same holds for the “many flows” scenario (cf. Table 4.4.4 for CP-TCP/EPF and Table 5.2.3 for REM), where the average backlog is on average 53% larger in the case of REM. Additionally, the link utilization is lower than with Explicit Price Feedback. Thus, though REM has the advantage of using only a single bit for path price transport, it results in a certain degradation in performance parameters as well. Nonetheless, REM still performs very well and can achieve much higher utilization at much lower average queuing delay than any of the conventional TCP variants.

Fairness

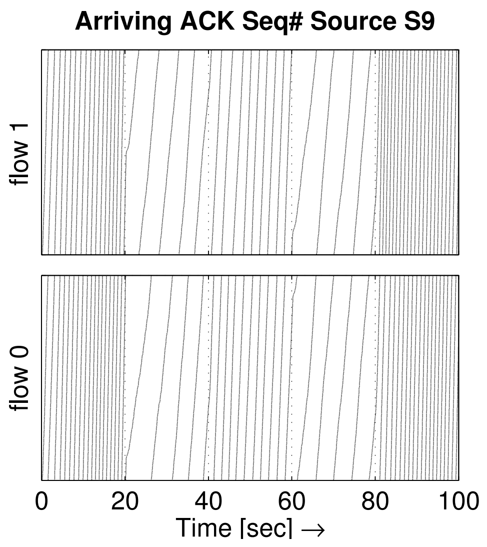
To evaluate fairness qualitatively, the acknowledgment sequence number plots shown in Figure 5.2.7 are used. These plots display the increasing sequence numbers of the arriving acknowledgments over time (cf. Subsection 4.4). Since additional flows (not shown) are activated during certain periods (cf. Figure 4.4.1b), the share of bandwidth that each flow receives and thus the slope of the parallel lines changes over time.

The plots can be used to evaluate fairness between different flows on the same path (here flow 0 and flow 1 are examined for each path displayed), to evaluate rate distribution qualitatively between flows on different paths (here four different paths are shown), and to evaluate qualitatively the rate and fairness changes when the network conditions change (i.e. change in the number of active flows over time). The source/sink pairs and the corresponding round-trip delays are shown in Table 4.4.1.

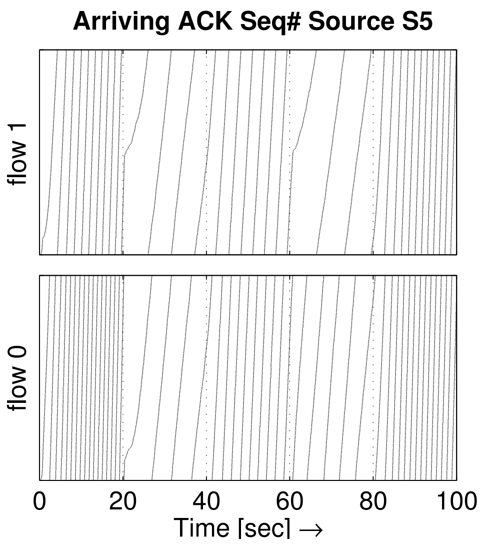
As expected from the proportional fairness criterion (3.2.22), the flows on the paths with the smallest network resource usage receive the highest rate (cf. Figure 4.4.1a), and the flows with the largest network resource usage receive the smaller rate (cf. Figure 4.4.1d). Also, both flows 0 and 1 of the same source receive the same rate. However, there is an exception: Simulations have revealed a few cases where a fair rate allocation between different flows from the same source cannot be established. An example is shown in Figure 5.2.7b: While flow 1 reduces its rate almost immediately when the additional flows are turned on at time 60 s, flow 0 only decreases its rate at a lower degree. This effect could only be observed in the “many flows” scenarios and is explained as follows: During the time interval [40–60 s] 400 flows are active. Then additional 400 flows are turned on. This causes the queues to fill up quickly as shown in Figure 5.2.6c. Since this significantly increases the round-trip time, the retransmission timer of almost all of the old flows expire before the acknowledgment is received. As a consequence, the congestion window is reduced to one. Only flow 0 of Figure 5.2.7b does not encounter a retransmission timeout and only slowly reduces the congestion window. Since the other 199 flows reduced their rates immediately, the congestion price decreases — allowing flow 0 to continue sending at a higher rate.

Conclusions

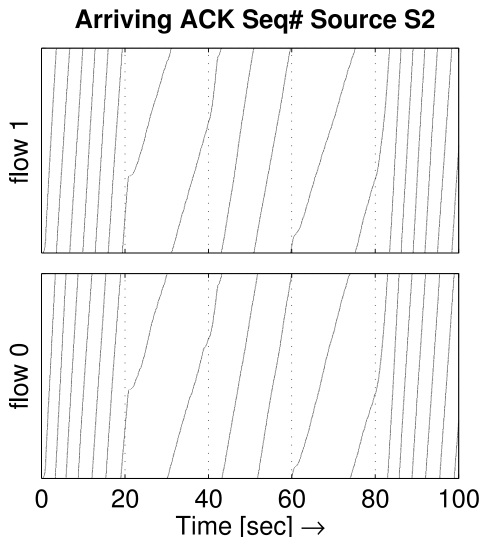
The idea of exponential marking allows REM to establish the correct average rates efficiently. However, as was demonstrated, the correct estimation of the path price is delayed because many packets are needed to estimate the marking probability. Together with the source algorithm, this



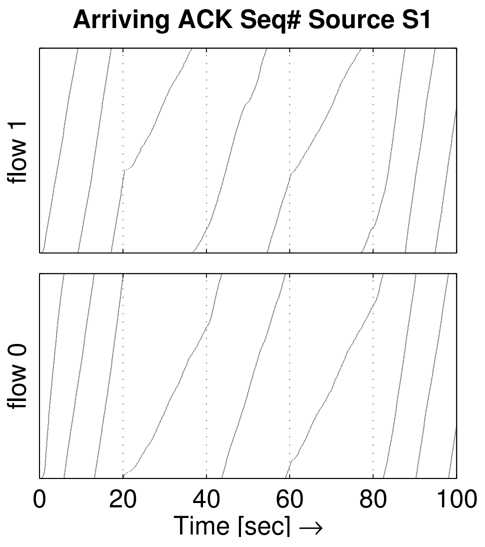
(a) Source 9



(b) Source 5



(c) Source 2



(d) Source 1

Figure 5.2.7: Acknowledgment sequence number plots (high number of competing flows) [ZHK01]

could lead to severe oscillations. Therefore, as the simulations have shown, REM parameters have to be carefully tuned for best performance. As mentioned initially, this is difficult for changing network conditions. Also with regard to the dynamic behavior, REM performs much worse than CP-TCP/EPF and VQM.

When the number of active flows is significantly increased, REM cannot maintain a link utilization above 90 % without adaptation of the parameters. Also, fairness between flows using the same resources is not always achieved. Nonetheless, REM is good approach to CP-based TCP that allows for service differentiation and can allocate rates better than TCP NewReno.

5.3 Summary of VQM and REM Simulation Results

Both REM and VQM are good candidates for Congestion Pricing based TCP implementations. They have different strengths and weaknesses, though. Similarly to CP-TCP/EPF, the Virtual Queue Mechanism (VQM) displays low queue sizes while maintaining high bottleneck link utilization. The congestion window also inversely follows the changes in load as is desired to keep utilization high and delay low. Even though VQM needs only a single bit for path price transport, it performs well. However, the rate allocation differs from the theoretical *weighted proportional fair* rate allocation. Thus, it is not possible to calculate rate distributions on a network. Even worse, flows that use more than one bottleneck link receive very small rate shares. If several bottleneck links are used, these connections can suffer from starvation. This is not tolerable in larger networks and therefore a very significant disadvantage of VQM.

REM, using its smart exponential encoding scheme, can establish the desired proportionally fair rate allocation very well. Multiple bottleneck links are not a problem, just like with CP-TCP/EPF. However, the estimation of the end–to–end marking probability introduces a significant delay in reaction to changed network conditions. Furthermore, REM has to be carefully tuned, as it tends to oscillate. VQM is easier to tune and shows smoother behavior leading to less fluctuating queue sizes. Both proposed implementations can be used with *service differentiation* by using different *willingness to pay* variables.

5.4 Single Bit Resource Marking (SBRM)

5.4.1 Motivation

In the previous sections two Congestion Pricing based TCP variants were introduced that only use a single bit to encode and transport the path price. They have different strengths and weaknesses (cf. Section 5.3). In this section, a new and practical TCP variant is proposed that combines the strengths of REM and VQM. This proposal is less complex and uses also only a single bit in the IP header, thus is perfectly compatible with the TCP Explicit Congestion Notification (ECN) [RFB01] extension. Since the name “Congestion Pricing” is often associated with payments, the name “Resource Marking” is preferred here. It still synonymously refers to the same Congestion Pricing framework that was presented in Chapter 3, but emphasizes that the framework is used for congestion control only and not for charging users.

5.4.2 Algorithms

Path Price Transport

A single bit is used to transport the *shadow prices*. Marking is exponential just like the REM proposal, thus the additive exponent property can be utilized to recover the *path price* (cf. Subsection 5.2.1).

Marking Algorithm

Since extensive simulations have shown that REM's three price computation rules PC1–PC3 on average do not differ significantly (cf. Subsection 5.4.4), for SBRM complexity is reduced as far as possible by basing the *marking algorithm* on the instantaneous queue size only.

Each packet leaving the queue is marked with the probability $m_l(t)$, where:

$$m_l(t) = 1 - \exp\left(-\gamma[b_l(t) - b_0]^+\right), \tag{5.4.1}$$

and γ is a scaling factor. To decrease feedback time, leaving packets are marked instead of entering packets. This will also cause the first packet of a burst of packets to be marked with a higher probability than the following packets. The underlying shadow price is calculated using the current queue size $b_l(t)$ minus a threshold value b_0, and can only be positive as indicated by the brackets $[]^+$.

Simulations conducted by the author of this dissertation have also shown that basing the marking on the instantaneous queue size only reduces oscillations significantly. However, this decision will cause a coupling of the congestion price and the queue length (cf. Section 2.4.1). Thus, if more flows share the link, the average queuing delay will increase. This will be demonstrated in Section 5.5. By choosing optimal parameter values, this undesired effect can be greatly reduced. Considering the strengths of SBRM, this drawback is acceptable. Furthermore, this property is common to most other proposed Active Queue Management (AQM) strategies such as RED, which will be also shown in Section 5.5.

Just like in other AQM strategies, the network operator can use the threshold value b_0 to set a desired target queue size. A high target queue size allows temporary bursty arrivals without punishing connections, and it will reduce times where the queue becomes totally empty even if there is still demand for bandwidth. This increases utilization of the following link. On the other hand, the payoff is an increased mean queue size and thus increased average queuing delay. It should be up to the network operator to chose a good value. For example, a router could have two queues: one for low latency traffic with a target queue size of zero, and one with a threshold $b_0 > 0$ to keep the utilization of the link close to 100%. b_0 could also be adapted dynamically, however, this is out of the scope of this investigation.

The marking algorithm (5.4.1) is similar to S. Athuraliya's and S. Low's Price Computation Rule 2 [AL00], but it differs in the use of the natural logarithm as fixed base for the exponential function. This eliminates the problem that a common ϕ must be agreed upon by all sources and network gateways. The simulations have shown that this value works well over a wide range of the number of bottleneck links and active sources. Here the scaling factor γ is chosen such that marking probability is 99% at maximum queue size. Using the two parameters b_0 and γ, the network operator can control the range in which the average queue size is located. However, as the application of control theory will show (cf. Chapter 6), it is a better choice to set $\gamma \approx \frac{1}{c}$.

Source Algorithm

The source algorithm of SBRM is based on the source algorithm (5.1.2)–(5.1.3) that was already used for VQM:

For every received non-marked acknowledgment, the congestion window is increased by:

$$\Delta cwnd = \overline{\kappa} \cdot \frac{RTT}{cwnd} \cdot w_n$$

and for every marked acknowledgment, differing from (4.2.3), the congestion window is decreased by:

$$\Delta cwnd = \overline{\kappa}.$$

As was described for VQM in Section 5.1, this algorithm was chosen for practical reasons because it also works with incorrect choices of the *willingness to pay* parameter w_n. Alternatively, a second variant of SBRM can be implemented that uses the original source algorithm (4.2.3). In both cases, the path price p_n is assumed to be either zero or one, depending on whether the packet was marked or not.

For the simulations described in the following subsections, TCP's congestion avoidance algorithm implemented in the UCB/VINT *Network Simulator 2 (ns-2)* [UCB] was modified to reflect SBRM's source algorithm. TCP's normal reaction to packet loss was unchanged.

5.4.3 Performance Evaluation (Simulations)

Rate Allocation

Again, the achieved rate allocation is evaluated using a double bottleneck link topology that was already described in Subsection 4.3.1. Since SBRM uses a random component, each simulation was repeated 30 times using different seeds to be able to calculate confidence intervals. In Figure 5.4.1, the resulting mean rates and 95% confidence intervals are shown. Again, the horizontal lines in the plots show the theoretical values for a weighted proportionally fair rate allocation. Ideally, the actually achieved values should be on the horizontal lines. For SBRM, there is no perfect match, but the resulting rate allocation is close to the desired weighted proportionally fair rate allocation. Additionally, a bottleneck link utilization is established that is on average higher than for any other evaluated TCP variant (cf. Table 5.4.1).

Table 5.4.1: Bottleneck link utilization

TCP variant	Utilization of core link 1	Utilization of core link 2	Average core link utilization
TCP NewReno+drop-tail	96.5 %	87.5 %	92.0 %
TCP NewReno+RED	95.1 %	89.9 %	92.5 %
CP-TCP/EPF	93.4 %	92.6 %	93.0 %
VQM	94.1 %	87.8%	90.9%
REM	94.9%	93.2%	94.1%
SBRM	97.9%	92.0%	95.0%

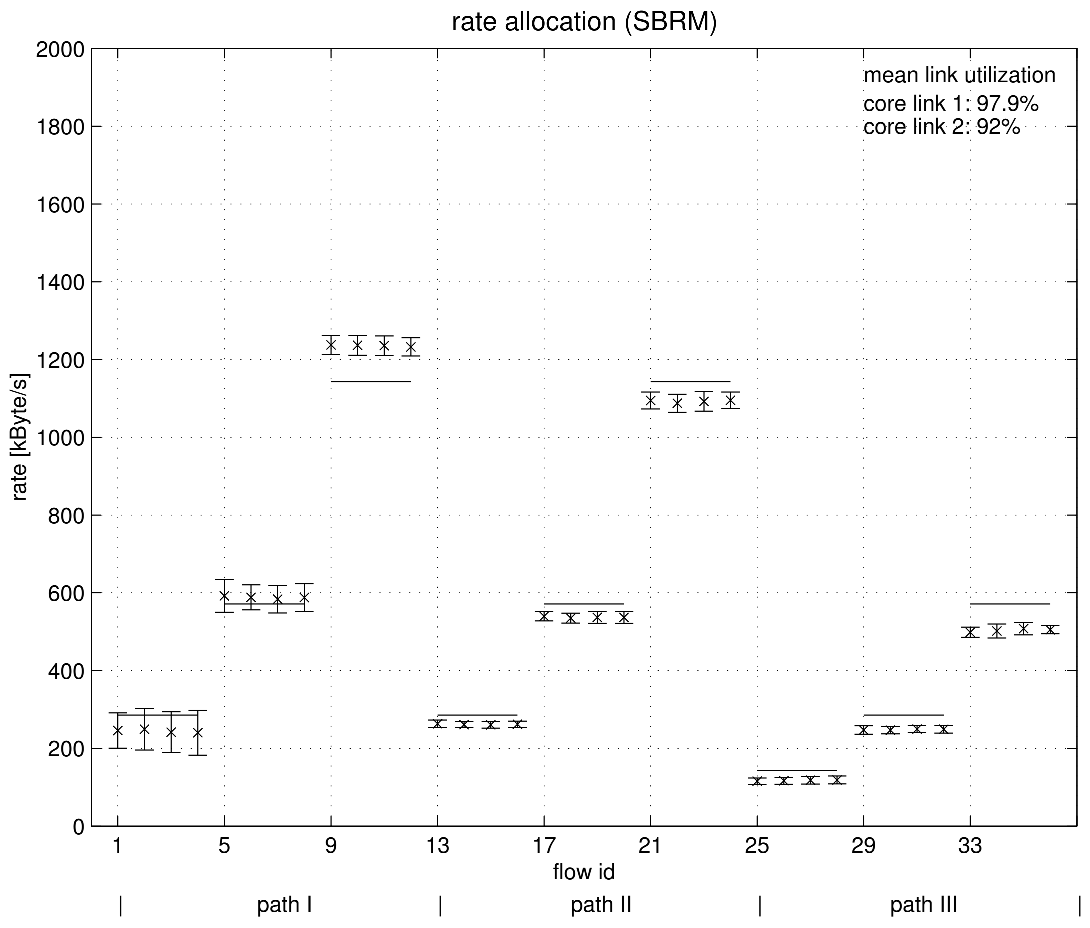


Figure 5.4.1: Rate allocation using SBRM [ZK02]

Also, service classes are established using three different *willingness to pay* settings. In each service class are four flows. As shown in Figure 5.4.1, they approximately receive the same bandwidth. Thus, fairness within a service class is achieved as desired. Although the rate allocation established by SBRM is not as close to the targets as REM (cf. Figure 5.2.1), it is much better than VQM (cf. Figure 5.1.1).

Dynamics

To evaluate dynamic behavior of SBRM, the number of active flows was again changed over time (cf. Subsection 4.3.1). The results of these simulations are shown in Figures 5.4.2 and 5.4.3. Figure 5.4.2 shows the resulting congestion windows for flows 1, 5, and 9.

The congestion window is changed inversely proportional to the load, which is the desired behavior. It is also more stable, and the oscillations seen with REM (cf. Figure 5.2.2) are not present any more. As a consequence, the development of the queue size (cf. Figure 5.4.3) remains almost constant at a low level. However, the average queue size is slightly higher than with VQM (cf. Figure 5.1.3) because of the introduction of a target queue size b_0. On the other hand, SBRM achieves the highest average bottleneck link utilization in comparison to any other TCP variant (cf. Table 5.4.1). Although the bottleneck link utilization of REM is close, it suffers from strong fluctuations of the queue size (cf. Figure 5.2.3). The behavior of the queue

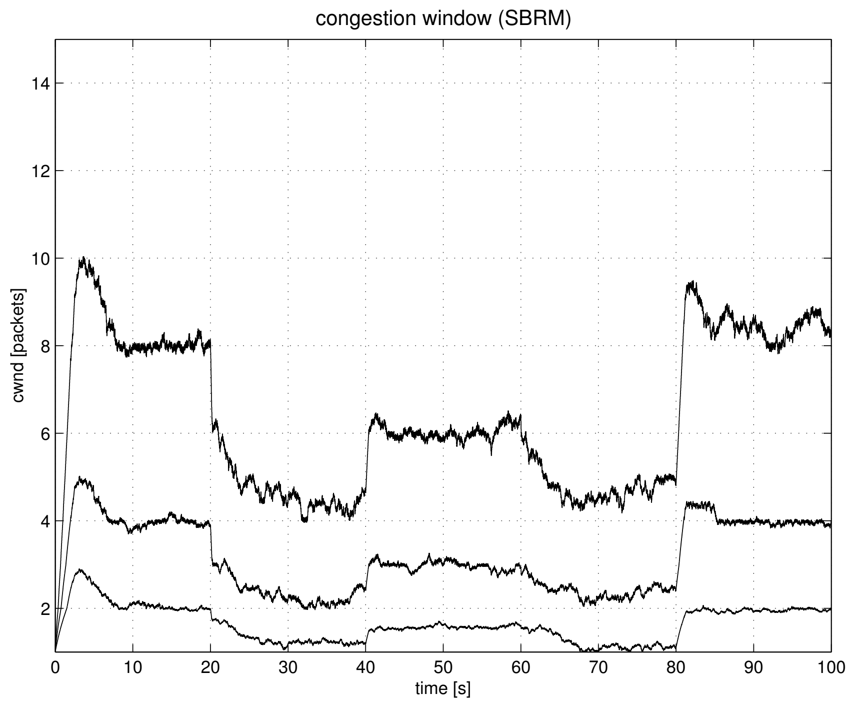


Figure 5.4.2: Congestion window (flows 1, 5, 9) [ZK02]

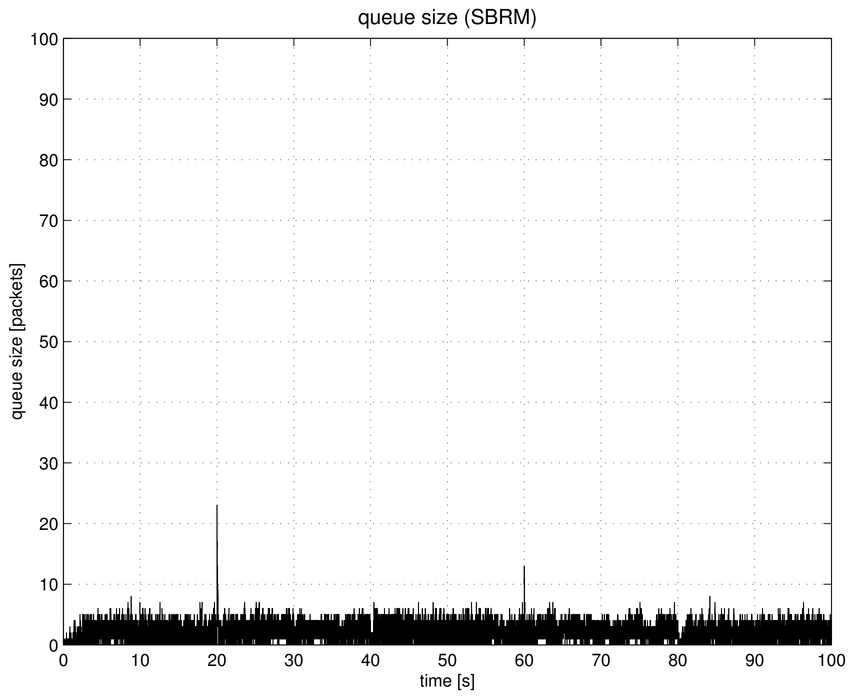


Figure 5.4.3: Queue sizes at core link of router R1 [ZK02]

with SBRM is even better than with CP-TCP/EPF (cf. Figure 4.3.7). In Figure 4.3.7, also conventional TCP variants are shown. They perform significantly worse than any Congestion Pricing based TCP variant.

5.4.4 Comparison of SBRM With Other Approaches in a Parking Lot Topology

Several different combinations of source, link, and path price transport algorithms were evaluated and compared to SBRM in [Ham01]. There, in a parking lot topology, the resulting throughput of the sources, the utilization of the links and the average backlogs were recorded for each combination for several simulation runs. Again, a scenario with a small number of flows ("few flows" scenario) and a scaled up scenario with increased capacities and a larger number of flows ("many flows" scenario) were used to evaluate scalability of the algorithms with regard to the number of flows (cf. Subsection 4.4.1). A summary of the results is shown in Figure 5.4.4. The displayed simulation type numbers are explained in Table 5.4.2.

The figures show significant variation of the resulting rate allocation in the "many flows" scenario for all single bit marking strategies (simulation types 1, 8 ,15, 22, and 23, cf. Figures 5.4.4a and b) with the exception of SBRM (simulation types 24, 25, and 26). SBRM, on the other hand, yields a rate allocation that is comparable to the strategies with full pricing information (simulation types 3, 4, 5, 7, 10, 11, 12, 14, 17, 18, 19, and 21). The utilization of the link is also well above 90% for SBRM at comparably low average backlogs (cf. Figures 5.4.4c and d). In summary, SBRM has proven to be the best Congestion Pricing implementation while using only a single bit for path price transport.

5.4.5 Conclusions

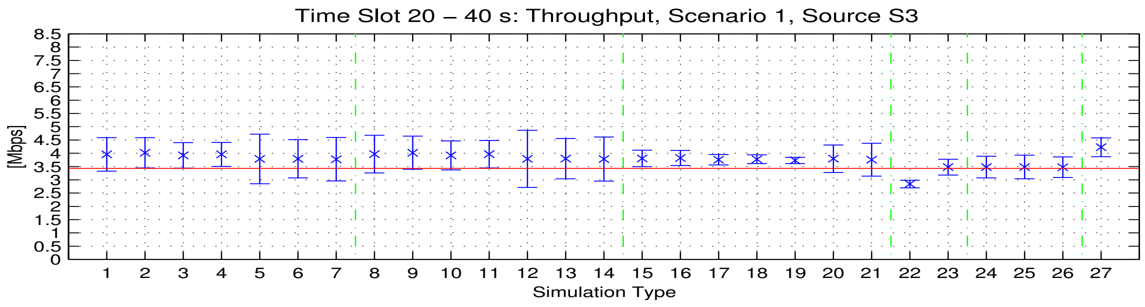
Single Bit Resource Marking (SBRM) is a simplified combination of the Direct Window Update source algorithm (4.2.3) and REM's price computation rule 2 (5.2.4) with some modifications to both algorithms. SBRM can establish the desired weighted proportionally fair rate allocation. Also, the dynamic behavior of the congestion window is as desired, leading to almost constantly low queue sizes and average bottleneck link utilization around 95%. SBRM is therefore a practical implementation of Congestion Pricing theory using only a single bit that performs comparably to CP-TCP/EPF. SBRM by far outperforms the conventional TCP algorithms as well (cf. Subsection 4.3.1). For these reasons, SBRM is the most promising approach to gain superior performance in a TCP/IP based network by using Congestion Pricing theory.

5.5 Steady-state Analysis of SBRM

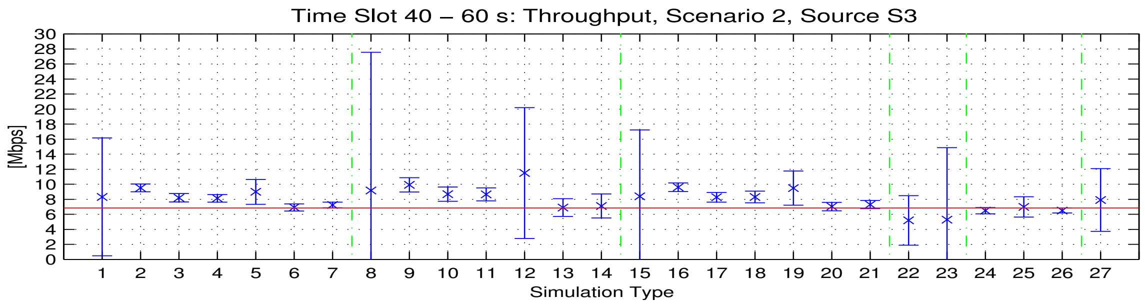
In this section a model will be presented for the analytical calculation of the steady-state of SBRM in a single bottleneck link scenario. This model can be used to calculate average rates, average backlog and average marking probability.

A single source solves the optimization problem stated by (3.2.8)–(3.2.9). Thus, in the steady-state, $U(x_n) - px$ will be at its maximum. This leads to the first order condition:

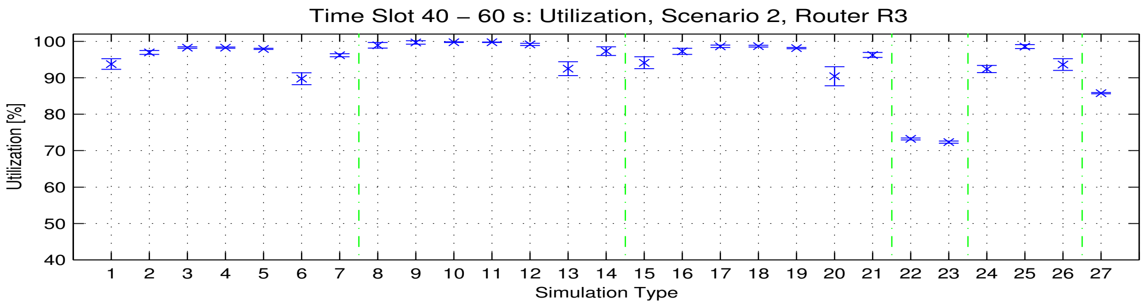
$$U'(x_n) = p_n^*. \qquad (5.5.1)$$



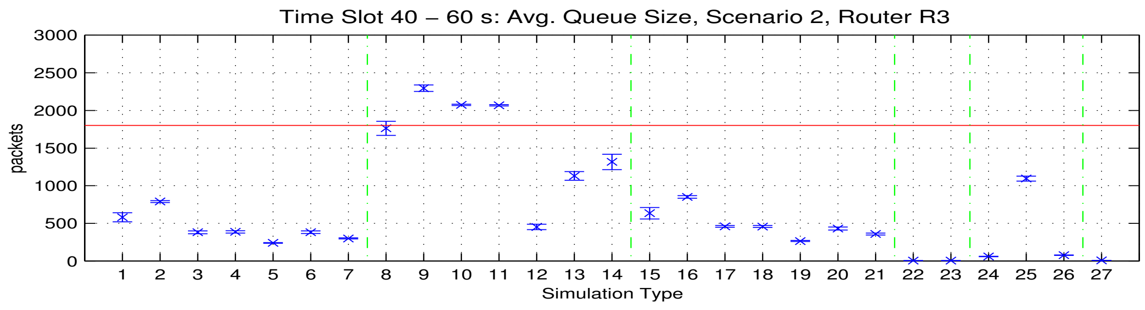
(a) Throughput of source 3 (8 flows)



(b) Throughput of source 3 (400 flows)



(c) Utilization of link 3 (400 flows)



(d) Average backlog of link 3 (400 flows)

Figure 5.4.4: Comparison of the resulting rate allocations of different combinations of source and link algorithms [Ham01]

Table 5.4.2: Legend to the simulation type numbers in Figure 5.4.4

Simulation	Source Algorithm	Link Algorithm	Rate Estimation	Slow Start
1	Vegas	PC1 1 Bit	RTT	yes
2	Vegas	PC1 4 Bit	RTT	yes
3	Vegas	PC1 EPF	RTT	no
4	Vegas	PC1 EPF	RTT	yes
5	Vegas	PC1 EPF	min. RTT	no
6	Direct Window Update	PC1 1 Bit	RTT	yes
7	Direct Window Update	PC1 EPF	RTT	no
8	Vegas	PC2 1 Bit	RTT	yes
9	Vegas	PC2 4 Bit	RTT	yes
10	Vegas	PC2 EPF	RTT	no
11	Vegas	PC2 EPF	RTT	yes
12	Vegas	PC2 EPF	min. RTT	no
13	Direct Window Update	PC2 1 Bit	RTT	yes
14	Direct Window Update	PC2 EPF	RTT	no
15	Vegas	PC3 1 bit	RTT	yes
16	Vegas	PC3 4 bit	RTT	yes
17	Vegas	PC3 EPF	RTT	no
18	Vegas	PC3 EPF	RTT	yes
19	Vegas	PC 3 EPF	min. RTT	no
20	Direct Window Update	PC3 1 Bit	RTT	yes
21	Direct Window Update	PC 3 EPF	RTT	no
22	Direct Window Update	VQM 1 Bit	RTT	yes
23	Direct Window Update	VQM n Bit	RTT	yes
24	Direct Window Update	SBRM + PC1	RTT	yes
25	Direct Window Update	SBRM + PC2	RTT	yes
26	Direct Window Update	SBRM + PC3	RTT	yes
27	TCP Reno	RED	n/a	yes

Since the utility function is strictly increasing, $p_n^* > 0$. Therefore $\sum_{l \in \mathcal{L}(n)} C_l(y_l) > 0$ and thus $b_l^* > 0$, where l is a bottleneck link. Because $b_l^* > 0$, the aggregate rate in the steady-state at link l cannot be less than the links capacity c_l. Since it also cannot be larger than the link's capacity, $y_l^* = c_l$. Therefore:

$$y_l^* = c_l = \sum_{n \in \mathcal{N}(l)} x_n^*. \tag{5.5.2}$$

In case of logarithmic utility functions (3.2.21),

$$U'(x_n) = \frac{w_n}{x_n}. \tag{5.5.3}$$

Inserting (5.5.1) and (5.5.3), (5.5.2) becomes:

$$c_l = \sum_{n \in \mathcal{N}(l)} \frac{w_n}{p_n^*}. \tag{5.5.4}$$

Now a single bottleneck link is assumed. From the *weighted proportional* fairness criterion (3.2.23), it is known that the available rate at the bottleneck link will be shared as follows:

$$x_n^* = \frac{w_n}{\sum_{i=1}^N w_i} \cdot c. \tag{5.5.5}$$

Thus, using (5.5.4) and (5.5.5) the average marking probability can be determined as:

$$m^* = p^* = \frac{w_n}{x_n^*} = \frac{\sum_{i=1}^N w_i}{c}. \tag{5.5.6}$$

It follows that $\sum_{n=1}^N w_n \leq c_l$, which was already discussed in Section 5.1.

The marking probability leads to the average queue length by using the inverse of (5.4.1):

$$b^* \quad = \quad b_0 - \frac{1}{\gamma} \ln(1 - p^*) \tag{5.5.7}$$

This reveals a problem: As the number of sources increases without adjustment of each source's willingness to pay w_n, the steady-state queue length will also increase. This problematic coupling of congestion and performance measure was already discussed in Subsection 2.4.1. But since the marking probability increases exponentially, linear increase in the number of sources will only lead to a logarithmic increase in queuing delay.

Using (2.3.1), the steady-state congestion window can finally be calculated as follows:

$$cwnd_n^* = RTT_m^* \cdot x_n^*. \tag{5.5.8}$$

Table 5.5.1 shows the steady-states in a single bottleneck link scenario for both TCP Reno with RED queues[2] and SBRM. The TCP Reno+RED model was taken from [Low00]. Again, the table shows that the resulting average rate depends on the round-trip time (RTT) only for TCP Reno.

[2]Assumption is made that the steady-state queue size is between minimum and maximum thresholds.

Table 5.5.1: Steady-states of TCP Reno with RED queues and SBRM in a single bottleneck link topology

	TCP Reno+RED [Low00]	SBRM
Average rate	$x_n^* = \frac{1}{RTT_n} \frac{1}{\sum_{i=1}^{N} \frac{1}{RTT_i}} c$	$x_n^* = \frac{w_n}{\sum_{i=1}^{N} w_i} c$
Mean marking probability	$p^* = \frac{2}{2+c^2 \left(\frac{1}{\sum_{n=1}^{N} \frac{1}{RTT_n}} \right)^2}$	$p^* = \frac{\sum_{n=1}^{N} w_n}{c}$
Mean queue length	$b^* = th_{min} + \frac{1}{\rho} p^*$	$b^* = b_0 - \frac{1}{\gamma} \ln(1-p^*)$

5.6 Compatibility with Conventional TCP

5.6.1 TCP-Friendliness

All flows on the Internet are supposed to be TCP friendly, i.e. under congestion they should behave similarly to conventional TCP variants. RFC2309 [BCC$^+$98] defines *TCP-compatible* flow as follows: “A *TCP-compatible* flow is responsive to congestion notification, and in steady-state it uses no more bandwidth than a conformant TCP running under comparable conditions (drop rate, RTT, MTU, etc.)”. Instead of using the term *TCP-compatible* flow, the term *TCP-friendliness* will be used here to refer to this feature. This term is also used in [FF99], where it is analogously defined as: “We say a flow is *TCP-friendly* if its arrival rate does not exceed the arrival of a conformant TCP connection in the same circumstances.”

Thus, according to these requirements, a newly proposed TCP variant such as SBRM or any other Congestion Pricing based algorithm should respond to congestion signals by reducing the transmission rate, and on average claim a bandwidth less or equal to the bandwidth claimed by “conformant TCP flows”. However, it is not clearly defined what a “conformant TCP flow” is. The conventional and common TCP variants Tahoe, Reno, and NewReno can vary significantly in the bandwidth shares they claim. But even if the assumption is made that all of the conventional TCP variants are “conformant”, the TCP-friendliness requirement is still debatable. The definitions both refer to “conformant TCP” as an upper bound for the mean bandwidth of a TCP-friendly flow. The authors of these definitions argue that TCP-friendliness is necessary to avoid congestion or even a congestion collapse. However, in the opinion of the author of this dissertation it is more important that a link is not overloaded and utilization is good. How the capacity is distributed between the competing flows should not be of concern from a congestion avoidance point of view. Thus, congestion could still be avoided even if flows claim more bandwidth that the conventional TCP variants. In [FF99], the authors additionally argue that taking more bandwidth than conformant TCP flows will also lead to unfairness: As the non-conformant flows claim more bandwidth, all other TCP-friendly flows have to reduce their share. While this is undoubtedly true, efficient usage of the network is more important than fairness. Especially current TCP variants are highly inefficient when the available bandwidth is very high.

In [FF99] the upper bandwidth of a "conformant TCP flow" in bytes per second is explicitly defined by the following formula:

$$BW \leq \frac{1.5\sqrt{\frac{2}{3}}B}{RTT\sqrt{p}},$$

where B is the maximum packet size in bytes, RTT is the round-trip time including queuing delays, and p is the aggregate packet drop/marking rate of the considered link. To achieve a sustained data rate of 10 Gbps with a RTT of 100 ms and a maximum packet size of 1500 Bytes, the packet drop probability must be less than $1.9 \cdot 10^{-10}$. This is roughly equivalent to one packet loss every $1\frac{2}{3}$ hours. It is commonly agreed that this is an unrealistic value. For this reason, TCP in high capacity networks is an active field of research. A modified TCP, called High-speed TCP, has been proposed to the IETF [Flo03].

SBRM, on the other hand, claims a bandwidth of:

$$BW = \frac{w_n B}{p},$$

where w_n is the *willingness to pay,* and p is the aggregate packet marking rate. This formula was derived from (5.5.5)–(5.5.6). To achieve a sustained transmission rate of 10 Gbps ($w_n = 1\frac{\text{pkts}}{\text{s}}$), the marking probability needs to be less than $1.1 \cdot 10^{-6}$. This is roughly equivalent to one congestion event every second. Figure 5.6.1 shows the bandwidth shares claimed by both variants as a function of the marking probability.

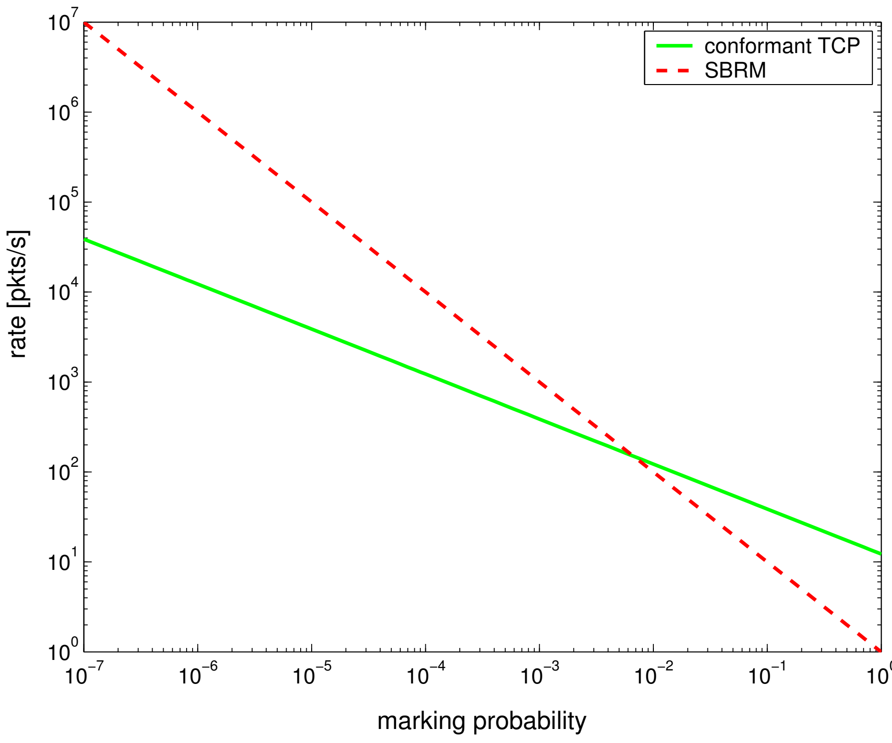


Figure 5.6.1: Claimed bandwidth by SBRM in comparison to "conformant TCP" ($RTT = 100\,ms$, $w_n = 1\,pkts/s$)

Depending on the choice of parameters, SBRM will claim a significantly higher rate than "conformant TCP" when the marking probability is low, and a lower rate when the marking probability is high. For this reason, SBRM is better able to achieve a high data rate in high capacity networks while it is even more conservative than conventional TCP variants in congested networks.

For these reasons, SBRM is a valid proposal although it violates the *TCP-friendliness* criterion. The impact on fairness between the Congestion Pricing based and conventional TCP variants is further examined in [Red02].

5.6.2 Compatibility with Existing TCP Sinks

Compatibility with current TCP is not an issue if Congestion Pricing based variants are being implemented in a private network. But even deployment in the global Internet is possible using slight modifications. Here it is assumed that ECN will be fully deployed on the network. For ECN-incapable networks, the conventional packet drop mechanism will be used. A future TCP stack should thus be able to interpret both types of congestion signals. SBRM as presented before is not 100% compatible with current Internet standards. Therefore some minor modifications are presented here in order to yield 100% compatibility.

Most of the Internet users receive more data than that what they send. This is also the assumption behind asymmetric digital subscriber lines (ADSL) that are becoming very popular. Additionally, most of the communications on the Internet are client/server based. Many clients (most users) request data from a few servers (web servers, mail servers, file servers). For this reason, to gain advantage from Congestion Pricing based TCP variants, it is sufficient to just migrate the servers. However, this is only possible if the Congestion Pricing based TCP variants are fully functional with conventional TCP sinks. Since the *Single Bit Resource Marking (SBRM)* proposal makes use of Explicit Congestion Notification (ECN) which is standardized by the IETF, it is inter-operable with ECN-capable routers and sinks. However, the original ECN standard [RFB01] requires that the TCP receiver echoes the *ECN Echo (ECE)* flag in every acknowledgment until it receives a *Congestion Window Reduced (CWR)* flag. It will take one round-trip time from echoing the first ECE flag until reception of the CWR flag. If no *Delayed-ACK* algorithm is used [JB88], for every received packet the receiver will send one acknowledgment. Thus, all acknowledgments will have the ECE flag set for one full congestion window if one packet was marked. The reason for this algorithm was to avoid problems when an acknowledgment gets lost.

SBRM must be modified to be compatible with this way of ECE signaling. Only minor modifications are necessary, but to distinguish this fully TCP-compatible variant of SBRM, it will be referred to as "TCP/RM" (TCP Resource Marking).

5.6.3 TCP/RM (Modified SBRM)

Source Algorithm

TCP/RM only modifies TCP's congestion avoidance algorithm when Explicit Congestion Notification (ECN) is used. Otherwise it is identical to TCP NewReno. Also, TCP's normal reaction to packet loss is unchanged.

For every received non-marked acknowledgment (no ECE flag set), the congestion window is increased by:

$$\Delta cwnd = \overline{\kappa} \cdot \frac{RTT}{cwnd} \cdot w_n, \tag{5.6.1}$$

and for the first marked acknowledgment (ECE flag set) per round-trip time, the congestion window is decreased by:

$$\Delta cwnd = \overline{\kappa} \cdot cwnd\,. \tag{5.6.2}$$

Since ECN receivers continue to echo the ECE flag until a CWND flag is received, the sender must ignore ECE flags for one round-trip time after it has reduced its congestion window. All parameters are identical to those of SBRM (cf. Section 5.4).

Round-Trip Time Estimation

Since the source algorithm directly depends on the estimate of the current round-trip time (RTT), it is important that the estimate is good. Generally, TCP's original smoothed RTT estimate can be used for this purpose. However, special care has to be taken with retransmissions. Karn's algorithm [KP87] forbids the use of RTT measurements of retransmitted packets because it is not possible to determine whether the acknowledgment refers to the original or to the retransmitted segment. However, Karn's algorithm does not solve the problem with the timing of segments that were sent *after* the lost segment but before recovery and which thus will produce duplicate acknowledgments. Such a segment can only be acknowledged when the retransmitted segment has been received. This will introduce an additional delay which is not part of the actual round-trip time. Therefore the author of this dissertation proposes to discard RTT measurements of retransmitted segments and all following segments that were sent before the lost packet was detected. To avoid this problem and for better accuracy, it is recommended to use TCP time-stamp option [JBB92].

Marking Algorithm

The SBRM marking algorithm as presented in Section 5.4 is used. However, any marking algorithm such as RED [FJ93] could also be used, TCP/RM will still work.

5.6.4 TCP/RM Simulations and HTTP Model

In the previous sections the rate allocation and dynamics of SBRM were analyzed with only greedy sources. In this section the TCP/RM implementation is verified by using more realistic traffic patterns. For this reason a single bottleneck link topology (cf. Figure 5.6.2) is considered and implemented in the Ptolemy Classic simulator [PCB]. Ten bidirectional flows are installed on each of the three paths: S1/S4, S2/S5, and S3/S6. Link capacities and delays are shown in Figure 5.6.2. The minimum round-trip time (RTT) of the flows going from source node S1 to S4 is 20 ms, from node S2 to S5 is 36 ms, and from S3 to S5 is 56 ms. Parameters for the TCP/RM sources are $\overline{\kappa} = 0.1$ and $w_n = 20$. The SBRM queue scaling factor γ is chosen such that marking probability is 99% at the queues maximum capacity.

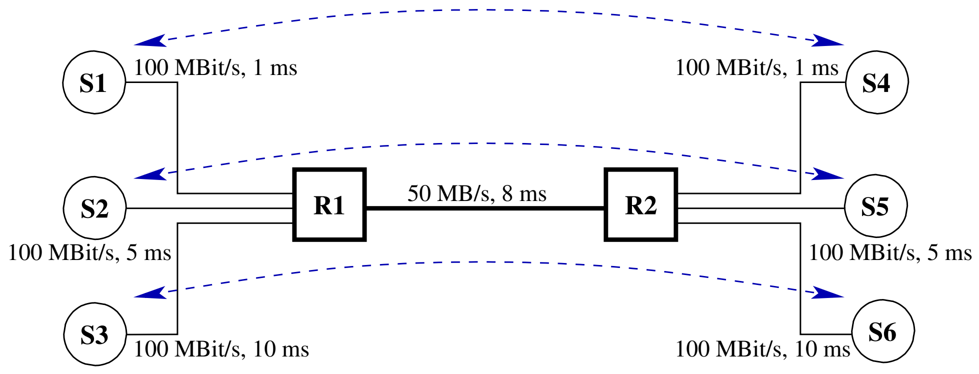


Figure 5.6.2: Single bottleneck link topology

Greedy model with SBRM queues

In the first experiment, greedy sources are used. The routers R1 and R2 use SBRM marking queues with a capacity of 300 packets and the marking threshold b_0 set to 30 packets. Simulated time is 60 seconds.

Since ECN marks are used as a less strict congestion signal than packet drops in TCP/RM, the congestion windows change less drastically with TCP/RM than with NewReno-ECN (cf. Figure 5.6.3). In Figure 5.6.4, histograms of the aggregated congestion windows of all flows on a path are shown. The leftmost columns are caused by the TCP receivers that keep their congestion window at one segment size. In the case of TCP/RM, the histogram is more narrow which suggests more stable congestion windows than with NewReno-ECN. It also implies better fairness between flows on the same path. However, for fairness between flows on different paths, the congestion windows should be significantly larger on the path S3-S6 than on the path S1-S3 because of the greater round-trip time. The results are not as good as was shown with SBRM [ZK02]. This is due to the low resolution (100 ms) of the smoothed round-trip time estimate in the Ptolemy Classic Simulator. Figure 5.6.5 shows the histogram of the queue size at

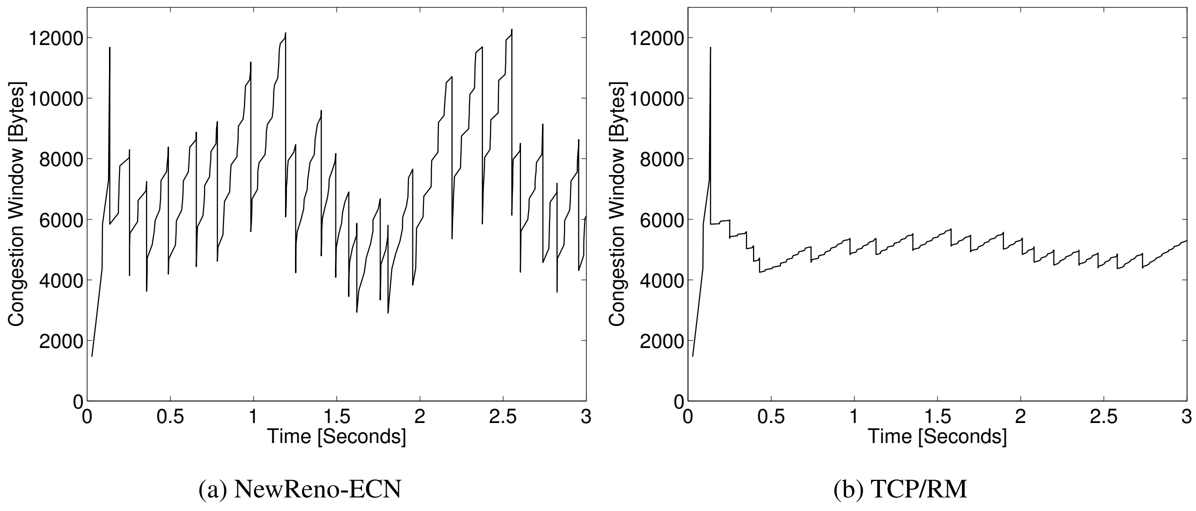
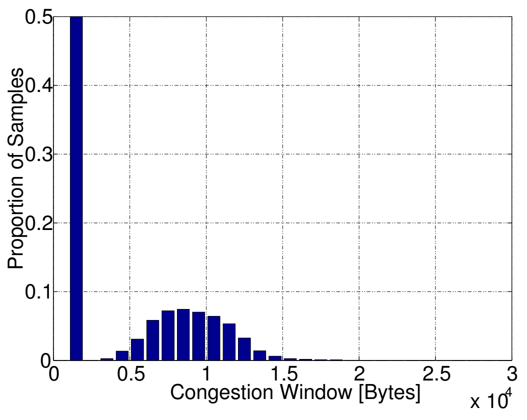
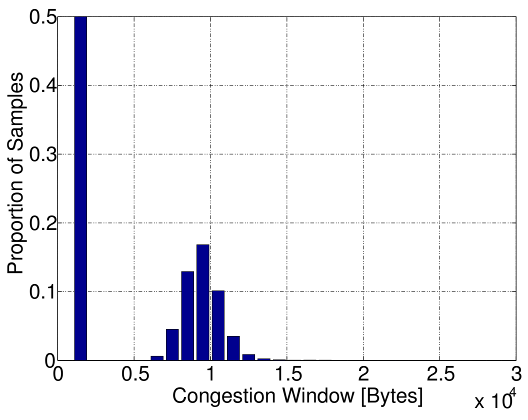


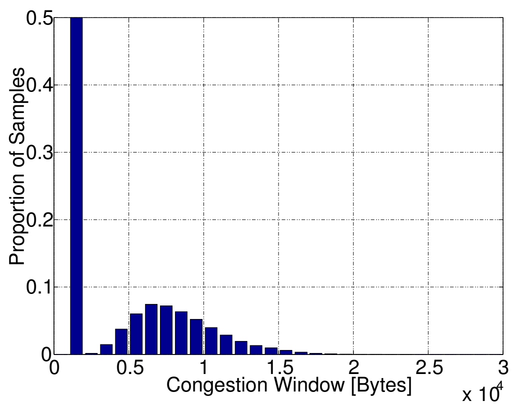
Figure 5.6.3: Run of the congestion windows (S2-S5)



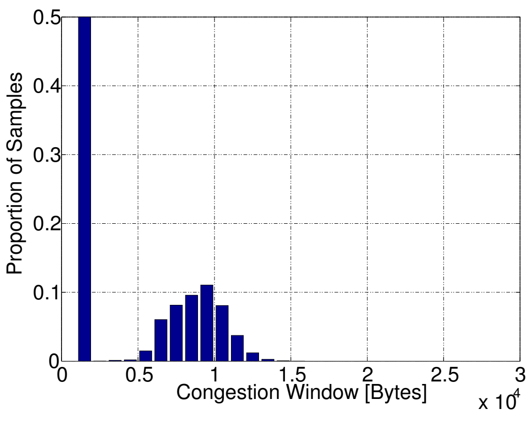
(a) NewReno-ECN, path S1-S4



(b) TCP/RM, path S1-S4



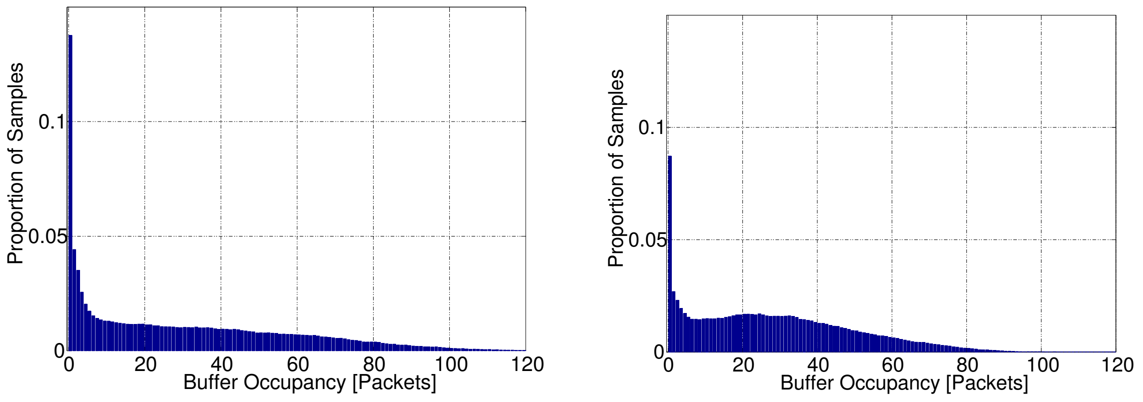
(c) NewReno-ECN, path S3-S6



(d) TCP/RM, path S3-S6

Figure 5.6.4: Greedy model with SBRM queues: histograms of congestion windows

the SBRM marking queue at router R1 serving the bottleneck link. It can be seen from the figure



(a) NewReno-ECN (b) TCP/RM

Figure 5.6.5: Greedy model with SBRM queues: histograms of buffer occupancies at R1

that on average the buffer occupancy is less variable and slightly lower with TCP/RM sources than with TCP NewReno-ECN sources. Usually, choosing smaller queue sizes by adjusting active queue management parameters leads to a trade-off with regard to link utilization. But even with slightly lower average queue occupancy, bottleneck link utilization is significantly higher with TCP/RM (cf. Table 5.6.1).

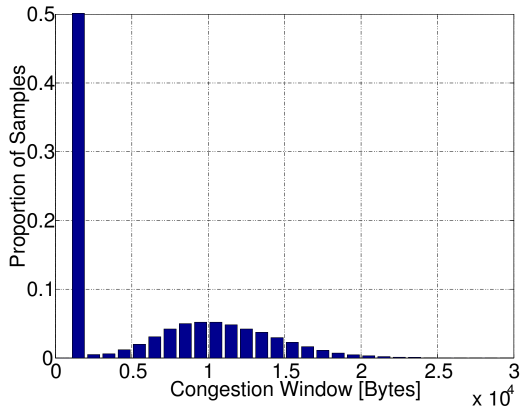
Table 5.6.1: Link utilization and mean buffer occupancy (R1)

	NewReno-ECN	TCP/RM
Link utilization	90%	94%
Mean buffer occupancy	30.0 packets	27.7 packets
Mean queuing delay	7.2 ms	6.6 ms

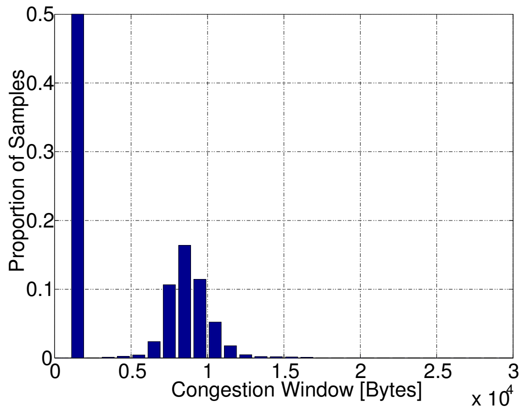
Greedy model with RED queues

The SBRM marking queue is replaced by a RED queue to examine the behavior of TCP/RM in networks that do not use the SBRM queues. Parameters for the RED queues are $th_{min} = 30$, $th_{max} = 90$, thus setting the desired queue size around 60. p_{max} is 0.2, and $q_w = 0.02$. Maximum capacity of the queue is 300 packets.

As with SBRM queues, the histogram of the congestion windows is narrower in the case of TCP/RM (Figure 5.6.6). However, the histogram of the buffer occupancy using TCP/RM has a local maximum at the desired equilibrium queue size of 60 packets (Figure 5.6.7). Also bottleneck link utilization is slightly higher for TCP/RM. In this case it comes with the trade-off of a higher mean queuing delay (Table 5.6.2).

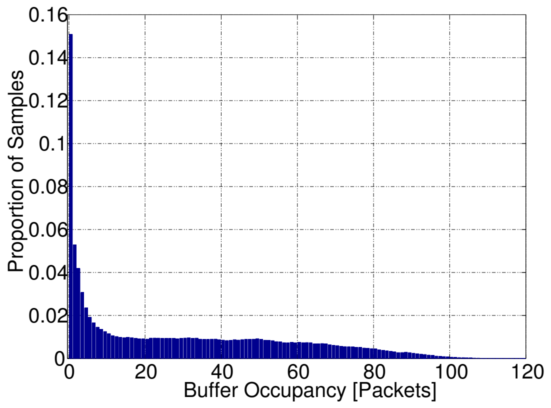


(a) NewReno-ECN, path S1-S4

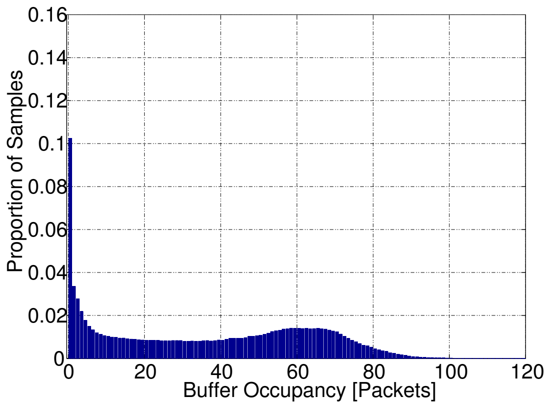


(b) TCP/RM, path S1-S4

Figure 5.6.6: Greedy model with RED queues: histograms of congestion windows



(a) NewReno-ECN



(b) TCP/RM

Figure 5.6.7: Greedy model with RED queues: histograms of buffer occupancies at R1

Table 5.6.2: Link utilization and mean buffer occupancy (R1)

	NewReno-ECN	TCP/RM
Link utilization	89%	92%
Mean buffer occupancy	28.8 packets	35.8 packets
Mean queuing delay	6.9 ms	8.6 ms

HTTP model with SBRM queues

For realistic traffic patterns, a HTTP model is used here that was presented in [BK02]. It uses a geometric distribution for the number of HTTP-objects per request (mean 6.55 objects), and a truncated power-tail distribution for the objects' sizes (mean 10 kBytes, minimum 100 Bytes, maximum 10 MBytes). The mean off-time of the sources is 0.5 seconds. The number of flows is increased to 200 per path to compensate for the off-times. Simulated time is 60 seconds.

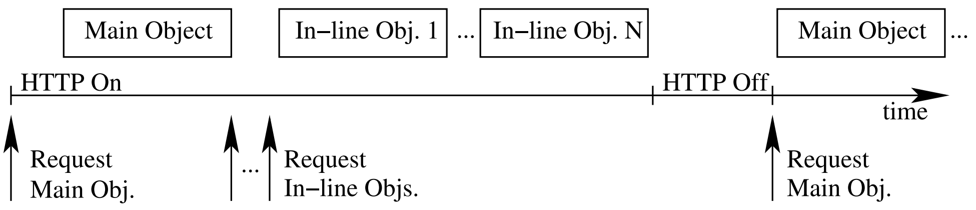
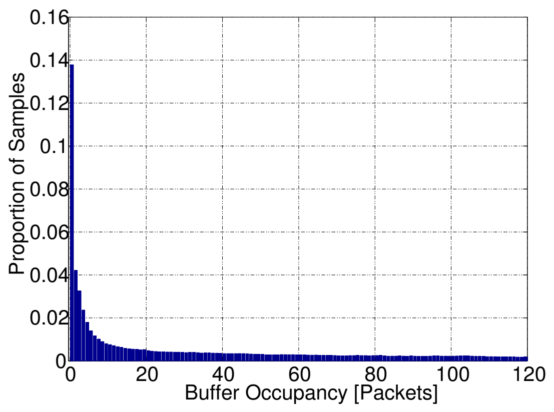
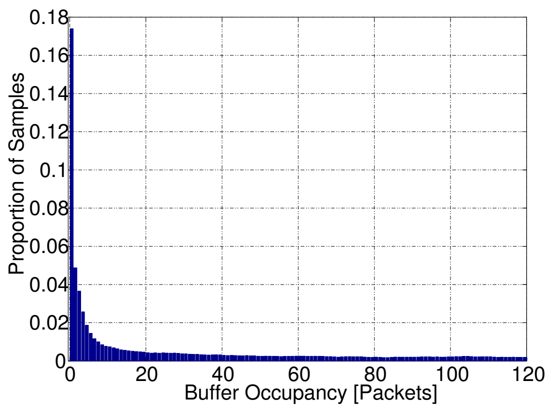


Figure 5.6.8: HTTP model [Bel03]

In Figure 5.6.9 the histograms for the buffer occupancy are shown for TCP NewReno-ECN and TCP/RM. Although mean buffer occupancy is slightly smaller with TCP/RM (cf. Table 5.6.3), both histograms look alike. The main reason is the effect of the slow start algorithm. Since HTTP flows often last only for a few packets, congestion avoidance mode is often never entered. Since both variants use the same slow start algorithm, they behave similarly. But still, TCP/RM experiences fewer fast retransmits and — more important — fewer timeouts (Table 5.6.3). The mean throughput value shown is the inverse of the mean time needed to transfer an average web page including all embedded objects from the server to the user.



(a) NewReno-ECN



(b) TCP/RM

Figure 5.6.9: HTTP model: histograms of buffer occupancies at R1

Conclusions

In this section a practical way was presented to implement Congestion Pricing theory in TCP in a backward compatible way using ECN. It was also shown that the modified TCP outperforms

Table 5.6.3: Link utilization and mean round-trip time

	NewReno-ECN	TCP/RM
Link utilization	89%	85%
Mean buffer occupancy	95.8 packets	91.2 packets
Mean queuing delay	23.0 ms	21.9 ms
Mean throughput	15 kBytes/s	14 kBytes/s
Fast retransmits	1042	1032
Timeouts	6590	6438

TCP NewReno-ECN in a single bottleneck link scenario for greedy FTP-like traffic. With realistic HTTP-like traffic, TCP/RM and TCP NewReno-ECN perform similarly. This is a result of HTTP connections being usually much shorter than file downloads. TCP connections then often never reach congestion avoidance mode. Thus, a modification of the slow start algorithm also has to be taken into consideration. For HTTP traffic, the transmission rate must quickly converge to the theoretical share. Oscillations are thus of less importance. One proposal is to implement Quick-Start [JF02]. Alternatively, the author of this dissertation suggests to use SBRM with an initially large $\bar{\kappa}$, which is then slowly reduced to the optimal value to increase stability.

5.7 Conclusions

Although SBRM and TCP/RM use only a single bit for path price transport, the rate allocation is close to the desired *weighted proportionally fair* rate allocation. Also, there is only little variation of the resulting rate for different simulation runs. To achieve a link utilization above 90%, the average backlog necessary can be chosen such that it is smaller than achievable with any other single bit marking variant. Thus, while the performance of SBRM and TCP/RM is close to the variants that use Explicit Price Feedback, SBRM and TCP/RM are compatible with current networks that allow only one bit for path price transport. SBRM and TCP/RM also allow the assignment of priorities of flows by choosing the *willingness to pay* parameter. This allows the designation of priority traffic such as business critical or interactive applications. SBRM and TCP/RM perform significantly better than any of the conventional TCP variants. SBRM and TCP/RM can therefore be used to solve almost all of the problems of conventional TCP that were mentioned in Subsection 2.4.1:

The transmission rate is only reduced after a segment has been lost due to overload.

Since Congestion Pricing information is used to signal overload, loss of segments is not required to signal congestion. Further, the simulations have shown that the average queue size is much lower than with conventional TCP variants at comparable utilization of the bottleneck link. Thus, overflow of the queue is much less likely and can in most cases completely be avoided.

Lost segments have to be retransmitted, thus causing a reduced goodput.

Since losses should only occur because of data corruption, retransmission is required, but much less likely. Also, the necessity for retransmission is decoupled from network load.

The detection of lost segments is slow or requires a minimum congestion window size.

Since every segment is to be acknowledged, no minimum congestion window size is necessary to use marking. Further, the congestion notification needs in most cases less than one round-trip time.

Most TCP versions do not recover well from multiple packet losses.

Since packet losses are very unlikely with SBRM, the chances of multiple packet losses are highly unlikely. Special handling of multiple packet losses is therefore not necessary.

TCP cannot distinguish between losses due to congestion and due to transient errors

Since SBRM leads to low queue sizes, queue capacities can be designed such that packet losses due to overflow are extremely unlikely. Since long before such overflow happens all packets are marked to indicate congestion, no special interpretation of the packet loss is necessary. SBRM could be designed in such a way that lost segments are just retransmitted without reduction of the transmission rate. This is necessary in high bandwidth or wireless networks where reduction of transmission rate after packet loss would be the wrong reaction.

The rate allocation depends on the round-trip time.

SBRM leads to a *weighted proportionally fair* rate allocation which is in general independent of round-trip time.

Deterministic drops may lead to the global synchronization problem.

SBRM marks probabilistically, thus avoiding global synchronization.

Direct coupling of the packet loss probability or queue size and the congestion measure is problematic.

The queue size and the congestion measure (markings) are also coupled in case of SBRM. Thus, this drawback cannot be solved by SBRM. Nonetheless, by adjusting the parameter γ, the grade of coupling can be influenced.

TCP and drop-tail queues will lead to full queues and high variance.

Since SBRM begins to signal congestion after the queue size becomes larger than the threshold b_0, sources begin to reduce their transmission rates at this point. Thus, SBRM leads to a queue size around that threshold which can be selected by the operator to ensure nearly perfect bottleneck link utilization while keeping average queuing delay low.

The optimal queue capacity is difficult to tune.

Queue capacity should be large enough to avoid packet losses. Since it is not used for signaling congestion, no tuning is necessary. Tuning can be achieved by changing the queue threshold parameter b_0, which can easily be automated.

TCP does not allow service differentiation or Quality of Service (QoS).

Since SBRM uses the *willingness to pay* parameter w_n, each user can select their preference for bandwidth. The bandwidth allocation is *weighted proportionally fair* and thus giving flows with higher willingness to pay higher rates. This feature can be used for service differentiation or relative QoS (i.e. Class of Service).

Chapter 6

Control Theoretic Analysis

In this chapter control theory is applied to the Congestion Pricing framework. Using a control theoretic model, scalability of the implementations can be examined in terms of delay, number of sources and link capacity. Here a control theoretic model is presented for SBRM that can be used to further understand the algorithm and for further improvement.[1]

6.1 Motivation

Historically, Internet protocols were designed by network specialists and not by control theoreticians. Although the cycle of packet marks or drops and rate adjustments can be viewed as a closed-loop feedback system, it was believed that the system is too complex for theoretical analysis. Design of protocols and queue management algorithms was mostly done by means of simulation. Later, the development of fluid flow models for TCP and the Congestion Pricing framework allowed the analytical evaluation of the network protocols on the flow level. The algorithms presented in the previous chapters can for example be used to calculate the equilibrium rates, packet marking probability, queue sizes, and delay (cf. Section 5.5). Using these values, rate allocation and fairness at the steady-state can also be evaluated. This resulted in the possibility to view the rate control problem as an optimization problem. Using these models, the goodput and the use of network resources can be optimized. However, because the algorithms in the fluid-flow models are still very complex, their usefulness is limited. Especially the evaluation of stability with consideration of delays on the network only became possible recently with new studies on the application of control theory to fluid flow models [HMTG01a, LPW$^+$02, SLD02, WP02, FPL03]. The common idea behind the application of control theory is the determination of stability of the network at the steady-state. When observing queue sizes, very often oscillations at low frequencies can be seen that suggest some form of instability. These oscillations not only lead to jitter, but also to under-utilization of the link when the queue runs empty. Using the results from the application of control theory, boundary conditions for parameters of already existing protocols can be determined, and fully scalable algorithms can be developed in the future. For example, in [HMTG01a, LPW$^+$02, SLD02] it was shown that current TCP variants in combination with RED queues will lead to unstable networks when the round-trip delay grows beyond a certain bound, or even when the capacity of the links becomes very large. These findings are extremely important because both causes

[1]This chapter is an extended version of [ZK04].

for instability are very likely in future networks as network capacity can be expected to grow further. The growing importance of wireless communications — especially over long distances or even interstellar communications — will also introduce additional delays to the network. Generally, a network protocol should be designed in such a way that only parameter settings which can be controlled by the network operator could lead to instability.

6.2 Limitations of Control Theoretic Models

Control theoretic models for networks are still an active area of research. A major difficulty is the complexity of the models. For example, there is a loop dependency of the delay: The reaction of every source depends on the delay which in turn depends on all the sources' reactions on the same path. In [LPW$^+$02] it was shown that earlier control models that assumed the delay to be constant [HMTG01a, HMTG01b] yield incorrect results. Furthermore, as the algorithms involved are very complex, linearization around the equilibrium state is used. Thus, these models can only be used to evaluate linear stability around the stable operation point.

There is also a controversy among researches on how important *stability* is for networks and whether a stable operation point can be maintained at all. Although instability leads to oscillations on the network and thus to jitter and degradation in throughput, the network may still be in operable condition. This will be demonstrated in the following sections. Additionally, network conditions are changing constantly as new connections are established and older ones are discontinued. Thus, the stable operation point is constantly moving. Even worse, as every connection is a separate control system, a larger network basically consists of thousands and millions of control loops. All of these loops depend on the other loops, and thus a drift away from a desired operating point could have many causes. Nonetheless, stability is a requirement for the applicability of fluid-flow models that are used to optimize network usage. Even with these difficulties, application and examination of control theoretic models is very important. In Section 6.7 the impact of instability on performance parameters such as utilization and jitter will be demonstrated.

Additionally, to realize optimal stability, the rate adjustments of the sources must depend on the distance from the optimal operation point and the number of reacting sources (cf. Figure 2.4.2). However, this information is usually not available in current networks as the number of sources is not conveyed to every source. Moreover, single bit marking strategies can only indicate the direction of the required rate adjustment and not the amount. To achieve optimal stability, explicit price feedback has to be used or any other analog measure that gives the necessary information. While other researchers have started to design stable algorithms that either use additional information not yet available in IP based networks [KHR00], make use of delay information like TCP Vegas [FAS03], or use an exponential encoding scheme of the explicit price [ALLY01, LPW$^+$02], a simpler algorithm is implemented here where only the direction of the rate adjustment is known. The author of this dissertation opines that a sub-optimal solution can be accepted in favor of compatibility with current protocols because small oscillations around the steady-state at the packet level are not harmful and cannot be avoided in an active network with changing conditions. Furthermore, all control theoretic models are based on fluid-flow models and do not consider oscillations at the packet level. Thus, to some extent oscillations are normal, and therefore their existence must be accepted. By introducing this limitation, algorithms can be designed that are compatible with existing IP networks and only require changes on the server side. For example, as was shown in the previous chapter, SBRM is completely compatible with existing TCP receivers (cf. Subsection 5.6.3).

S. Low's FAST (Fast AQM Scalable TCP) [FAS03] on the other hand is an approach that also is compatible with existing IP networks and that uses an analog measure to convey the distance from the equilibrium point to the sources. S. Low picked up on the idea of using the delay as a congestion measure, which was first introduced with the TCP Vegas proposal [BOP94]. Thus, FAST changes the amount of the rate adjustment depending on the distance from the equilibrium point. However, arguments that have been brought against TCP Vegas also hold for FAST: In order to measure the congestion, the round-trip time without queuing delays must be known. Usually, the minimum measured round-trip time is used for that. If the route changes and the minimum round-trip time becomes greater, the algorithm will fail [LWA99, MLAW99]. In current wired networks, a change of route within a connection's duration seems to be somewhat unlikely. But with wireless access technology becoming more important in the future, roaming will be a significant issue. Unfortunately, roaming usually involves variation of propagation delays.

For these reasons it is difficult to design a stable TCP variant that will perform optimally in every scenario. With SBRM a variant was chosen that will perform sub-optimally, but well in most network conditions and always better than the current TCP implementations.

6.3 Method

In Chapter 4 the Congestion Pricing based TCP algorithms (source, link, and price feedback algorithms) were derived using the differential equations that were the result of the Congestion Pricing framework (cf. Chapter 3). These equations were then further used in a fluid-flow model to determine the steady-state (cf. Section 5.5). The control theoretic model is also based on these equations. However, since the fluid-flow model is very complex, the differential equations will be linearized around the equilibrium point. By transforming these differential equations to the Laplace domain, transfer functions can be found for every part of the network. The Nyquist criterion will be applied to the transfer function model to determine stability.

Figure 6.3.1 shows a block diagram of a single component of a control system. For example,

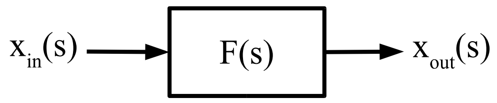


Figure 6.3.1: Block diagram of a component of a control system in Laplace domain

the SBRM queue algorithm could be modeled as such a component. Assuming continuous time and a fluid flow model, the instantaneous queue length would be modeled as input $x_{in}(s)$. The SBRM link algorithm $F(s)$ then transforms this input into a marking probability $x_{out}(s)$.

A control system commonly includes a negative feedback. This is displayed in Figure 6.3.2. The transfer function that describes the closed-loop control system can be determined as fol-

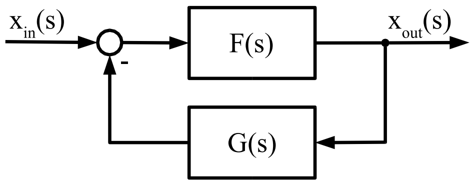


Figure 6.3.2: Block diagram of a closed loop control system

lows:

$$\begin{aligned} L_{closed}(s) := \frac{x_{out}(s)}{x_{in}(s)} &= \frac{(x_{in}(s) - x_{out}(s)G(s))\,F(s)}{x_{in}(s)} \\ &= \frac{F(s)}{1 + F(s)G(s)}. \end{aligned}$$

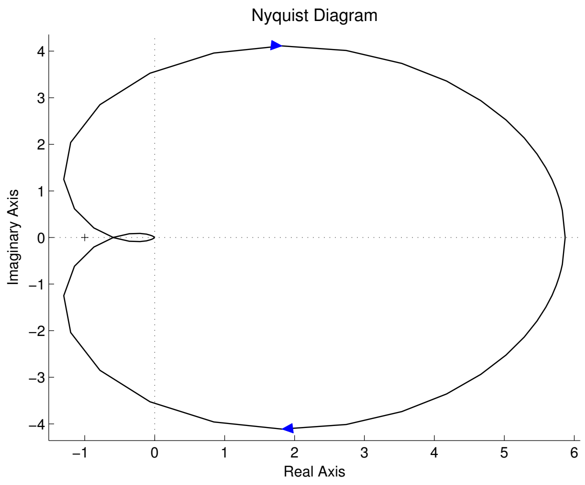
The corresponding open-loop transfer function is obtained by cutting the loop at any point. Thus if one cuts to the right of $G(s)$:

$$\begin{aligned} L_{open}(s) := \frac{x_{out}(s)}{x_{cut}(s)} &= \frac{-x_{cut}(s)G(s)F(s)}{x_{cut}(s)} \\ &= -F(s)G(s) \end{aligned}$$

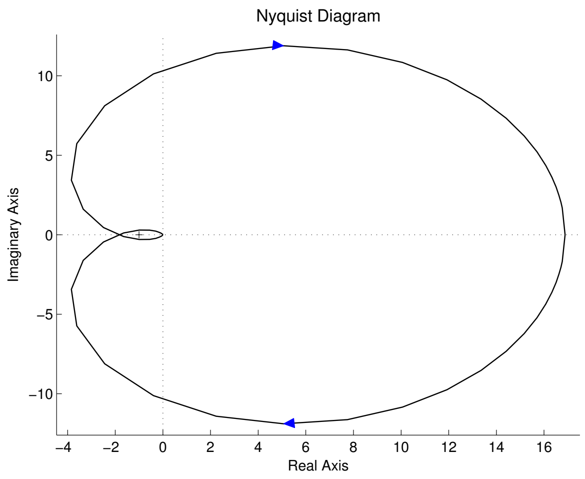
The zeros of the *characteristic equation* $1+F(s)G(s)$ are the poles of the closed-loop system, and its poles are the poles of the open-loop system.

The Nyquist diagram is the contour plot of the mapping of $L(s) := F(s)G(s)$ to the complex plane where $s = j\omega$. Note that this is the shifted plot of the characteristic equation. P is defined as the number of open-loop poles enclosed by the Nyquist contour, N as the number of clockwise encirclements of the $(-1,0)$ point minus the number of counter-clockwise encirclements of the $(-1,0)$ point, and Z as the number of real, positive poles of the closed-loop system. If Z is a positive, nonzero number, the closed-loop system is unstable (stability condition). For the systems discussed here, $Z = 0$. According to the Nyquist stability criterion, the closed-loop system is stable if and only if the Nyquist diagram of $L(s)$ encircles the $(-1,0)$ point in the complex plane P times in the counter-clockwise direction, thus if and only if $P+N = 0$. Since one can further assume all single components of the control system and therefore the open-loop $L(s)$ to be stable, the *characteristic equation* $1+F(s)G(s)$ has no poles in the right-hand plane, and thus $P = 0$. The Nyquist stability criterion then states in particular that the closed-loop system is stable if and only if the Nyquist diagram of $L(s)$ does not encircle the $(-1,0)$ point (cf. Figure 6.3.3) [Unb97]. Furthermore, the *critical frequency* f_{crit} is defined as the frequency at which the contour plot is closest to the $(-1,0)$ point.

Stability for the type of systems used here can also be evaluated in Bode diagrams. For the system to be stable, the magnitude (gain) in dB must be negative at the critical frequency, where the phase becomes -180°, and the phase must be greater than -180° at the frequency where the magnitude in dB changes from positive to negative values (cf. Figure 6.3.4). Both criteria are interchangeable. The gain and phase margins describe the distance of the actual values to the maximum values for the system to still be stable. The larger they are, the less likely the system will become unstable due to small changes. Using Bode diagrams, it is possible to evaluate the influence of different parameters on the stability of the closed-loop control system.



(a) Stable system



(b) Unstable system

Figure 6.3.3: Example Nyquist diagrams

6.4 Non-linear Model of SBRM

6.4.1 Window Evolvement under SBRM

For control theoretic analysis of SBRM, the source algorithm given by (4.2.3) and the marking algorithm given by (5.4.1) will be used. A fraction of $1 - p_n(t)$ of all incoming acknowledgments will increase the congestion window by

$$\Delta cwnd_n(t)_+ = \overline{\kappa} \cdot w_n \cdot \frac{RTT_n}{cwnd_n(t)},$$

where $p_n(t)$ is the end-to-end marking probability observed at source n, $0 < \overline{\kappa} < 1$ is a constant gain, w_n is the willingness to pay, RTT_n is the round-trip time, and $cwnd_n(t)$ is the current congestion window. Since acknowledgments are arriving at a rate of $x_n(t - RTT_n(t))$, where $x_n(t)$ is defined as rate of source n at time t:

$$x_n(t) = \frac{cwnd_n(t)}{RTT_n(t)}, \tag{6.4.1}$$

the rate and amount of increase equals

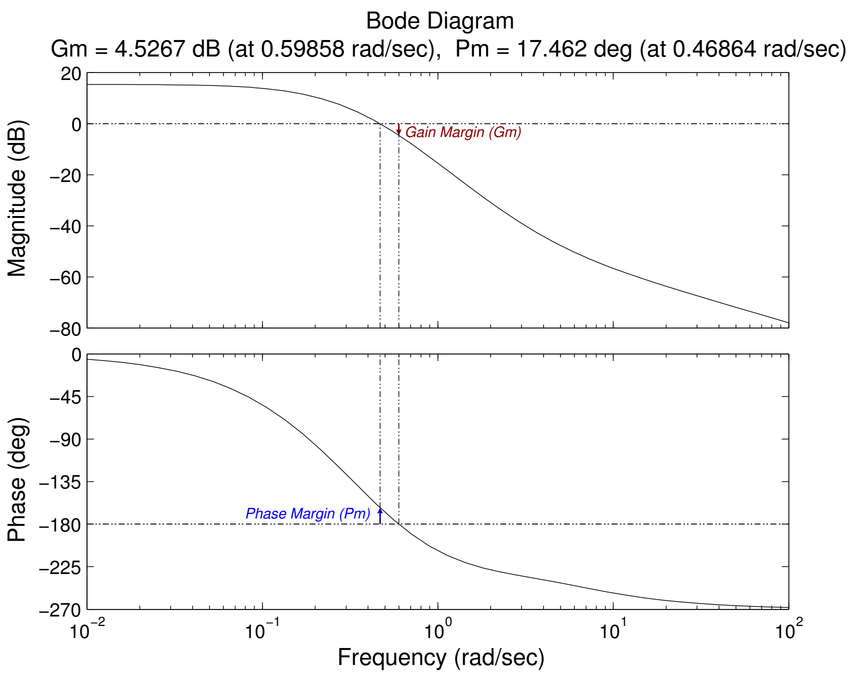
$$cwnd'_n(t)_+ = \overline{\kappa} \cdot w_n \cdot (1 - p_n(t)) \cdot \frac{RTT_n(t)}{cwnd_n(t)} \frac{cwnd_n(t - RTT_n(t))}{RTT_n(t - RTT_n(t))}.$$

Similarly, a fraction of $p_n(t)$ of all incoming acknowledgments will decrease the congestion window by

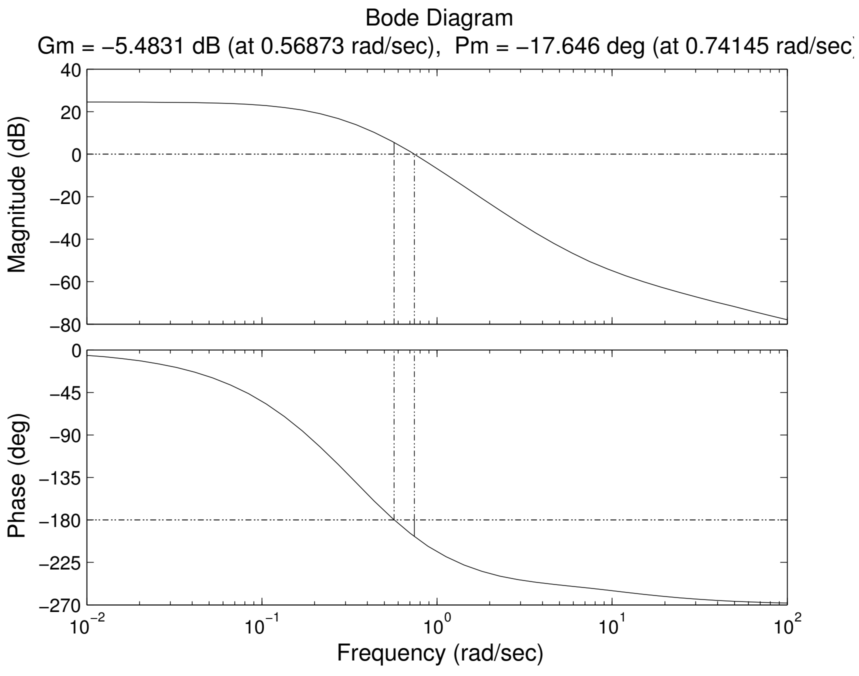
$$\Delta cwnd_n(t)_- = \overline{\kappa} \cdot \left(w_n \cdot \frac{RTT_n}{cwnd_n(t)} - 1 \right)$$

at a rate of

$$\frac{cwnd_n(t - RTT_n(t))}{RTT_n(t - RTT_n(t))} \cdot p_n(t).$$



(a) Stable system



(b) Unstable system

Figure 6.3.4: Example Bode diagrams

Thus the differential equation becomes:

$$cwnd_n'(t)_- = \overline{\kappa} \cdot \left(w_n \cdot \frac{RTT_n}{cwnd_n(t)} - 1 \right) \cdot p_n(t) \cdot \frac{cwnd_n(t - RTT_n(t))}{RTT_n(t - RTT_n(t))}.$$

Put together, congestion window under SBRM evolves according to:

$$cwnd_n'(t) = \overline{\kappa} \cdot \left(w_n \cdot \frac{RTT_n(t)}{cwnd_n(t)} - p_n(t) \right) \cdot \frac{cwnd_n(t - RTT_n(t))}{RTT_n(t - RTT_n(t))}. \tag{6.4.2}$$

Figure 6.4.1b shows a block diagram of SBRM's congestion control. In comparison, Figure 6.4.1a shows TCP Reno's *congestion avoidance* algorithm with Explicit Congestion Notification (ECN).

6.4.2 SBRM Queue

An SBRM queue marks packets at each link $l \in \mathcal{L}(n)$ with a probability $m_l(t)$, where $\mathcal{L}(n)$ is the set of all links that are used by source n. The end-to-end marking probability is then

$$p_n(t) = 1 - \Pi_{l \in \mathcal{L}(n)} \left(1 - m_l(t - \tau_{ln}^b(t)) \right),$$

where $\tau_{ln}^b(t)$ is the backward delay from link l to source n. Assuming that $m_l(t)$ is small, it is approximated:

$$p_n(t) \cong \sum_{l \in \mathcal{L}(n)} m_l(t - \tau_{ln}^b(t)). \tag{6.4.3}$$

All flows aggregate at link l to a load of

$$y_l(t) = \sum_{n \in \mathcal{N}(l)} x_n(t - \tau_{ln}^f(t)), \tag{6.4.4}$$

where $\tau_{ln}^f(t)$ is the forward delay from link l to source n. Thus $RTT_n(t) = \tau_{ln}^f(t) + \tau_{ln}^b(t)$ for any $l \in \mathcal{L}(n)$. It can be modeled as propagation delay d_n plus queuing delay of all queues on the path:

$$RTT_n(t) = d_n + \sum_{l \in \mathcal{L}(n)} \frac{b_l(t)}{c_l}, \tag{6.4.5}$$

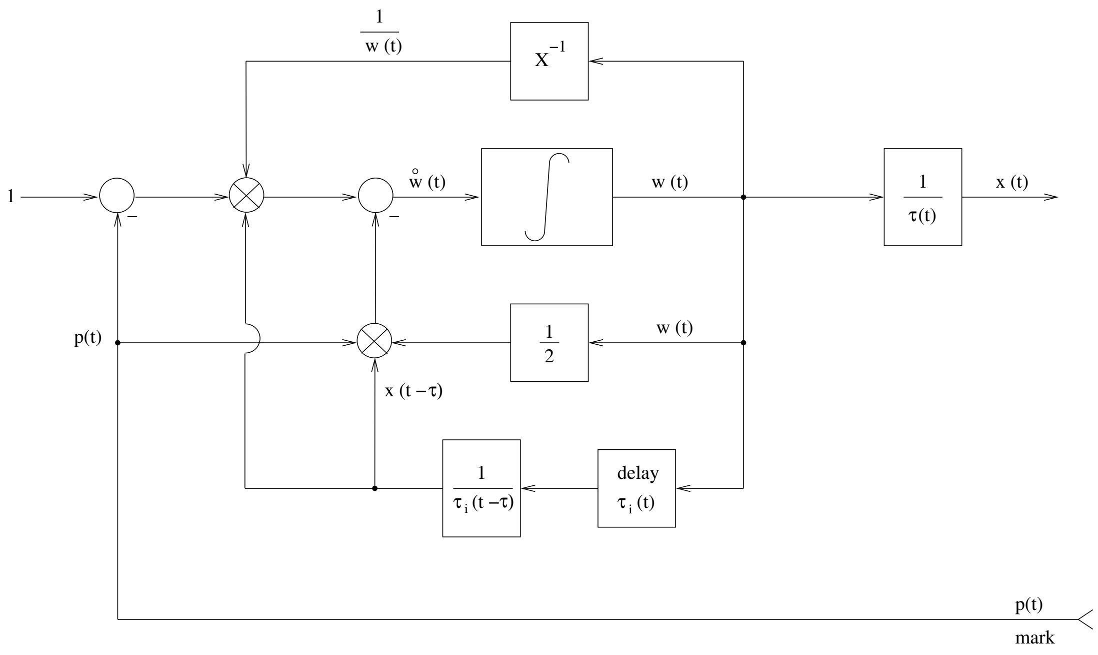
where c_l is the capacity of link l and $b_l(t)$ denotes the instantaneous queue size at link l. For $b_l(t) > 0$ the queue length evolves according to

$$b_l(t)' = y_l(t) - c_l. \tag{6.4.6}$$

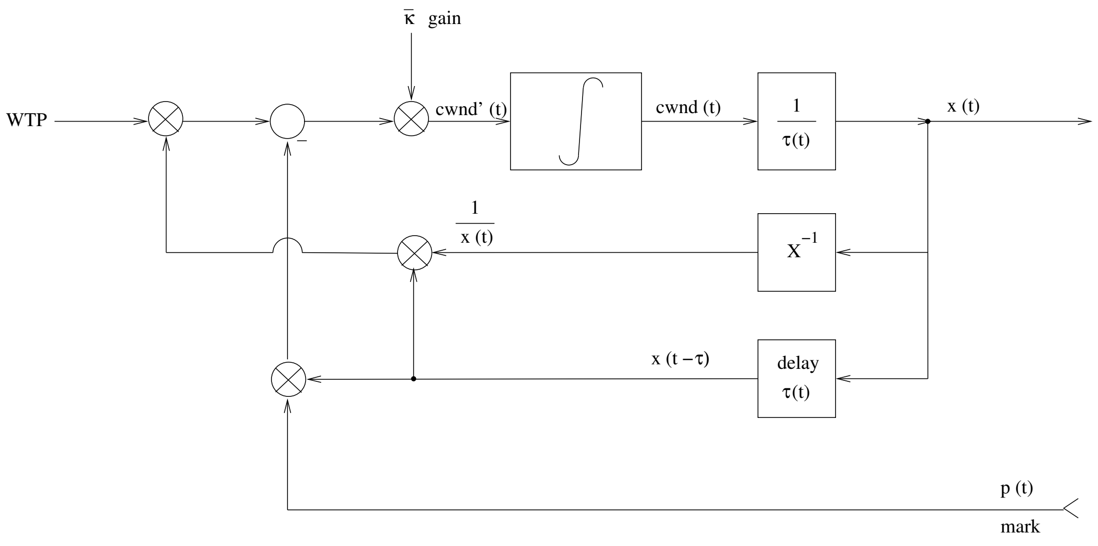
Each packet leaving the SBRM queue is marked with the probability:

$$m_l(t) = 1 - \exp\left(-\gamma [b_l(t) - b_0]^+\right), \tag{6.4.7}$$

where $\gamma > 0$ is a scaling factor [ZK02].



(a) TCP Reno with ECN (congestion avoidance)



(b) SBRM (source algorithm)

Figure 6.4.1: Block diagrams of TCP's congestion control (source algorithm)

6.5 Linear Model of SBRM

6.5.1 Linearization

These equations are now linearized to study its stability around equilibrium. The time-varying round-trip delay is approximated by its equilibrium value except for the window-rate relation (6.4.1), where the time-varying delay (6.4.5) must be used [LPW+02].

The steady-state is determined by $cwnd'(t) = b_l'(t) = 0$, thus from (6.4.2):

$$w_n \frac{RTT_n^*}{cwnd_n^*} = p_n^*.$$

This is also equivalent to (5.5.1). Variation around the equilibrium is then given by:

$$\begin{aligned}
\delta cwnd_n'(t) &= \bar{\kappa}_n \cdot \left(\frac{cwnd_n^*}{RTT_n^*} \cdot \left(w_n \cdot \frac{1}{cwnd_n^*} \cdot \delta RTT_n(t) - \right.\right. \\
&\qquad \left.\left. w_n \cdot \frac{RTT_n^*}{cwnd_n^{*2}} \cdot \delta cwnd_n(t) - \delta p_n(t) \right) + 0 \right) \\
&= \bar{\kappa}_n \cdot \\
&\qquad \left(\frac{w_n}{RTT_n^*} \cdot \left(\sum_{l \in \mathcal{L}(n)} \frac{\delta b_l(t - \tau_{ln}^f)}{c_l} \right) - \frac{w_n}{cwnd_n^*} \cdot \delta cwnd_n(t) - \frac{cwnd_n^*}{RTT_n^*} \cdot \delta p_n(t) \right) \qquad (6.5.1)
\end{aligned}$$

Aggregated link rates are linearized using (6.4.1), (6.4.4) and (6.4.5) [LPW+02]:

$$\begin{aligned}
\delta y_l(t) &= \sum_{n \in \mathcal{N}(l)} \delta x_n(t - \tau_{ln}^f(t)) \\
&= \sum_{n \in \mathcal{N}(l)} \left(\frac{\delta cwnd_n(t - \tau_{ln}^f)}{RTT_n^*} - \frac{cwnd_n^*}{RTT_n^{*2}} \cdot \sum_{k \in \mathcal{L}(n)} \frac{\delta b_k(t - \tau_{kn}^f)}{c_k} \right), \qquad (6.5.2)
\end{aligned}$$

where $\mathcal{N}(l)$ is the set of all sources that use link l, and where the round-trip time around the equilibrium is derived from (6.4.5):

$$\delta RTT_n(t) = \sum_{l \in \mathcal{L}(n)} \frac{\delta b_l(t)}{c_l}. \qquad (6.5.3)$$

Similarly, the buffer process is linearized: From (6.4.6):

$$\delta b_l'(t) \quad = \quad \delta y_l(t) \qquad (6.5.4)$$

Using (6.4.3) and (6.4.7), the linearized SBRM marking algorithm finally is described by:

$$\begin{aligned}
\delta m_l(t) &= \gamma_l \cdot e^{-\gamma_l([b_l^* - b_0]^+)} \cdot \delta b_l(t) \qquad (6.5.5) \\
\delta p_n(t) &= \sum_{l \in \mathcal{L}(n)} \delta m_l(t - \tau_{ln}^b) \qquad (6.5.6)
\end{aligned}$$

6.5.2 Linear Model in the Laplace Domain

The system described by differential equations (6.5.1–6.5.6) can also be written in the Laplace domain as:

$$\begin{aligned}
\delta cwnd_n(s) &= \sum_{l\in\mathcal{L}(n)} \frac{\overline{\kappa}\cdot\frac{w_n}{RTT_n^*}\cdot\sum_{k\in L(n)}\left(\frac{\delta b_k(s)}{c_k}\cdot e^{-\tau^f_{kn}s}\right)-\overline{\kappa}\cdot\frac{cwnd_n^*}{RTT_n^*}\cdot\delta m_l(s)\cdot e^{-\tau^b_{ln}s}}{s+\overline{\kappa}\cdot\frac{w_n}{cwnd_n^*}} && (6.5.7)\\
\delta y_l(s) &= \sum_{n\in\mathcal{N}(l)}\left(\frac{\delta cwnd_n(s)}{RTT_n^*}\cdot e^{-\tau^f_{ln}s}-\sum_{k\in\mathcal{L}(n)}\frac{cwnd_n^*}{RTT_n^{*2}}\cdot\frac{\delta b_k(s)}{c_k}\cdot e^{-\tau^f_{kn}s}\right) && (6.5.8)\\
\delta b_l(s) &= \frac{\delta y_l(s)}{s} && (6.5.9)\\
&= \frac{\sum_{n\in\mathcal{N}(l)}\frac{\delta cwnd_n(s)}{RTT_n^*}\cdot e^{-\tau^f_{ln}s}}{s+\sum_{m\in\mathcal{N}(l)}\sum_{k\in L(m)}\frac{cwnd_m^*}{RTT_m^{*2}}\cdot\frac{1}{c_k}\cdot\frac{\delta b_k(s)}{\delta b_l(s)}\cdot e^{-\tau^f_{km}s}} && (6.5.10)\\
\delta m_l(s) &= \gamma_l\cdot e^{-\gamma_l([b_l^*-b_0]^+)}\cdot\delta b_l(s) && (6.5.11)
\end{aligned}$$

6.5.3 Single Bottleneck Link Model with Heterogeneous Sources

The system of equations in the Laplace domain (6.5.7–6.5.11) is now simplified for a single bottleneck link model with heterogeneous sources:

$$\begin{aligned}
\delta cwnd_n(s) &= \frac{\overline{\kappa}\cdot\frac{w_n}{RTT_n^*}\cdot\frac{\delta b(s)}{c_k}\cdot e^{-\tau^f_n s}-\overline{\kappa}\cdot\frac{cwnd_n^*}{RTT_n^*}\cdot\delta m(s)\cdot e^{-\tau^b_n s}}{s+\overline{\kappa}\cdot\frac{w_n}{cwnd_n^*}} && (6.5.12)\\
&:= \delta cwnd_n^{delay}(s)+\delta cwnd_n^{mark}(s)\\
\delta y(s) &= \sum_{n\in\mathcal{N}}\left(\frac{\delta cwnd_n(s)}{RTT_n^*}\cdot e^{-\tau^f_n s}\right)-\\
&\quad\sum_{n\in\mathcal{N}}\left(\frac{cwnd_n^*}{RTT_n^{*2}}\cdot\frac{\delta b(s)}{c}\cdot e^{-\tau^f_n s}\right) && (6.5.13)\\
&:= \delta y^{rate}(s)+\delta y^{delay}(s)\\
\delta b(s) &= \frac{\delta y^{rate}(s)+\delta y^{delay}(s)}{s}=\frac{\delta y^{rate}(s)}{s-\frac{\delta y^{delay}(s)}{\delta b(s)}} && (6.5.14)\\
&= \frac{\sum_{n=1}^{N}\frac{\delta cwnd_n(s)}{RTT_n^*}\cdot e^{-\tau^f_n s}}{s+\sum_{m=1}^{N}\frac{cwnd_m^*}{RTT_m^{*2}}\cdot\frac{1}{c}\cdot e^{-\tau^f_m s}}\\
\delta m(s) &= \gamma\cdot e^{-\gamma([b^*-b_0]^+)}\cdot\delta b(s) && (6.5.15)
\end{aligned}$$

Using equations (6.5.12)–(6.5.15) the transfer functions is written as:

$$P_{source_n^{mark}}(s) = \frac{\delta cwnd_n^{mark}(s)}{\delta m_n(s)\cdot e^{-\tau_n^b s}} = -\frac{\overline{\kappa}\cdot\frac{cwnd_n^*}{RTT_n^*}}{s+\overline{\kappa}\cdot\frac{w_n}{cwnd_n^*}} \tag{6.5.16}$$

$$P_{source_n^{delay}}(s) = \frac{\delta cwnd_n^{delay}(s)}{\delta b_n(s)\cdot e^{-\tau_n^f s}} = \frac{\overline{\kappa}\cdot\frac{w_n}{RTT_n^*}\cdot\frac{1}{c}}{s+\overline{\kappa}\cdot\frac{w_n}{cwnd_n^*}} \tag{6.5.17}$$

$$P_{buffer}(s) = \frac{\delta b(s)}{\delta y^{rate}(s)} = \frac{1}{s+\frac{1}{c}\sum_{i=1}^{N}\frac{cwnd_i^*}{RTT_i^{*2}}\cdot e^{-\tau_i^f s}} \tag{6.5.18}$$

$$P_{aqm}(s) = \frac{\delta m(s)}{\delta b(s)} = \gamma\cdot e^{-\gamma([b^*-b_0]^+)} \tag{6.5.19}$$

Note that all of these transfer functions have poles on the left-hand plane, i.e. the real parts of the poles are negative. Thus, each component of the open-loop is stable [Unb97].

The complete control system is depicted in Figure 6.5.1.

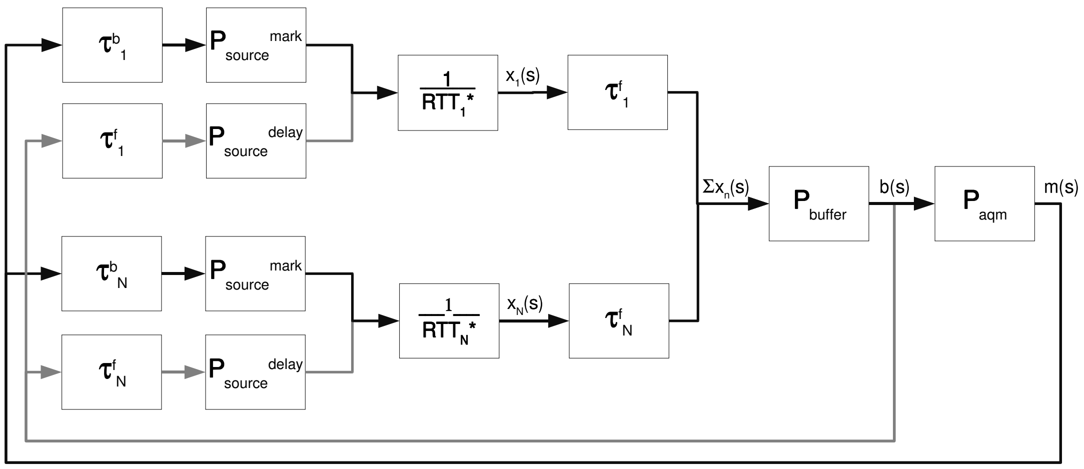


Figure 6.5.1: Single bottleneck link model with heterogeneous sources

The overall transfer function of the control system is now calculated:

$$\begin{aligned} F(s) &= \frac{\sum_{n=1}^{N}\left(e^{-\tau_n^b s}\cdot P_{source_n^{mark}}(s)\cdot\frac{e^{-\tau_n^f s}}{RTT_n^*}\right)\cdot P_{buffer}(s)}{1-\sum_{n=1}^{N}\left(P_{aqm}(s)\cdot e^{-\tau_n^b s}\cdot P_{source_n^{mark}}(s)\cdot\frac{e^{-\tau_n^f s}}{RTT_n^*}+e^{-\tau_n^f s}\cdot P_{source_n^{delay}}(s)\cdot\frac{e^{-\tau_n^f s}}{RTT_n^*}\right)\cdot P_{buffer}(s)} \\ &:= \frac{\sum_{n=1}^{N}G(s)}{1-\sum_{n=1}^{N}\left(P_{aqm}(s)\cdot G(s)+H(s)\right)} \end{aligned}$$

To determine stability of the system, the open-loop transfer function $L(s)$ must be determined. $L(s)$ is defined as:

$$\begin{aligned}
L(s) &:= -\sum_{n=1}^{N}\left(P_{aqm}(s)\cdot G(s)+H(s)\right)\\
&= -\sum_{n=1}^{N}\left(P_{aqm}(s)\cdot e^{-\tau_n^b s}\cdot P_{source_n^{mark}}(s)\cdot\frac{e^{-\tau_n^f s}}{RTT_n^*}+e^{-\tau_n^f s}\cdot P_{source_n^{delay}}(s)\cdot\frac{e^{-\tau_n^f s}}{RTT_n^*}\right)\cdot\\
&\quad P_{buffer}(s)\\
&= \sum_{n=1}^{N}\left(\gamma\cdot e^{-\gamma(b^*-b_0)}\cdot e^{-\tau_n^b s}\cdot\frac{\overline{\kappa}\cdot\frac{cwnd_n^*}{RTT_n^*}}{s+\overline{\kappa}\cdot\frac{w_n}{cwnd_n^*}}\cdot\frac{e^{-\tau_n^f s}}{RTT_n^*}-e^{-\tau_n^f s}\cdot\frac{\overline{\kappa}\cdot\frac{w_n}{RTT_n^*}\cdot\frac{1}{c}}{s+\overline{\kappa}\cdot\frac{w_n}{cwnd_n^*}}\cdot\frac{e^{-\tau_n^f s}}{RTT_n^*}\right)\cdot\\
&\quad \frac{1}{s+\frac{1}{c}\sum_{i=1}^{N}\frac{cwnd_i^*}{RTT_i^{*2}}\cdot e^{-\tau_i^f s}}\\
&= \sum_{n=1}^{N}\left(\frac{\overline{\kappa}\cdot cwnd_n^*\cdot e^{-RTT_n s}-\overline{\kappa}\cdot w_n\cdot\frac{1}{c}\cdot e^{-2\tau_n^f s}}{RTT_n^{*2}\left(s+\overline{\kappa}\cdot\frac{w_n}{cwnd_n^*}\right)}\right)\cdot\\
&\quad \gamma\cdot e^{-\gamma(b^*-b_0)}\cdot\frac{1}{s+\frac{1}{c}\sum_{i=1}^{N}\frac{cwnd_i^*}{RTT_i^{*2}}\cdot e^{-\tau_i^f s}}
\end{aligned}\tag{6.5.20}$$

The Eigenvalues are then $-\kappa\frac{w_n}{cwnd_n^*}$ and $-\frac{1}{c}\sum_{i=1}^{N}\frac{cwnd_i^*}{RTT_i^{*2}}e^{-\tau_i^f s}$. Since they are negative and nominators are non-zero, all components of the open loop are stable.

6.5.4 Single Bottleneck Link Model with Homogeneous Sources

Loop gain for a single bottleneck link model with homogeneous sources becomes:

$$L_{SBRM}(s)=N\cdot\frac{\overline{\kappa}\cdot cwnd^*\cdot e^{-RTTs}-\overline{\kappa}\cdot w\cdot\frac{1}{c}\cdot e^{-2\tau^f s}}{RTT^{*2}\left(s+\overline{\kappa}\cdot\frac{w}{cwnd^*}\right)}\cdot\gamma\cdot e^{-\gamma(b^*-b_0)}\cdot\frac{1}{s+\frac{N}{c}\frac{cwnd^*}{RTT^{*2}}\cdot e^{-\tau^f s}}.\tag{6.5.21}$$

Stability criteria can now be applied to this transfer function. As validation will prove, the Nyquist criterion is applicable.

For comparison, the corresponding loop gain for TCP Reno+RED is [LPW+02]:

$$L_{RED}(s)=N\cdot\frac{1}{RTT^*\cdot p^*\left(RTT^*s+p^*\cdot cwnd^*\right)}\cdot\frac{\alpha\cdot c\cdot\rho}{s+\alpha\cdot c}\cdot\frac{1}{s+\frac{N}{c}\frac{cwnd^*}{RTT^{*2}}\cdot e^{-\tau^f s}}\cdot e^{-\tau^f s},\tag{6.5.22}$$

where $0<\alpha<1$ is a RED parameter for calculation of the moving average of the queue size, and $\rho=\frac{p_{max}}{th_{max}-th_{min}}$ is derived from the remaining RED parameters.

6.6 Evaluation of the Control Theoretic Model

6.6.1 Evaluation of the Stability

Utilizing the derived loop gain for a single bottleneck link model with homogeneous sources (6.5.21), the stability can be determined for arbitrary sets of parameters. Using a set of parameters as shown in Table 6.6.1, the Nyquist diagram is plotted for several round-trip times from

Table 6.6.1: Parameters

TCP version	Parameter	Value
SBRM	$\overline{\kappa}$	0.2
SBRM	w	20
SBRM	γ	0.006
SBRM	b_0	50
Reno+RED	th_{min}	50
Reno+RED	th_{max}	550
Reno+RED	p_{max}	0.1
Reno+RED	$gentle$	$true$

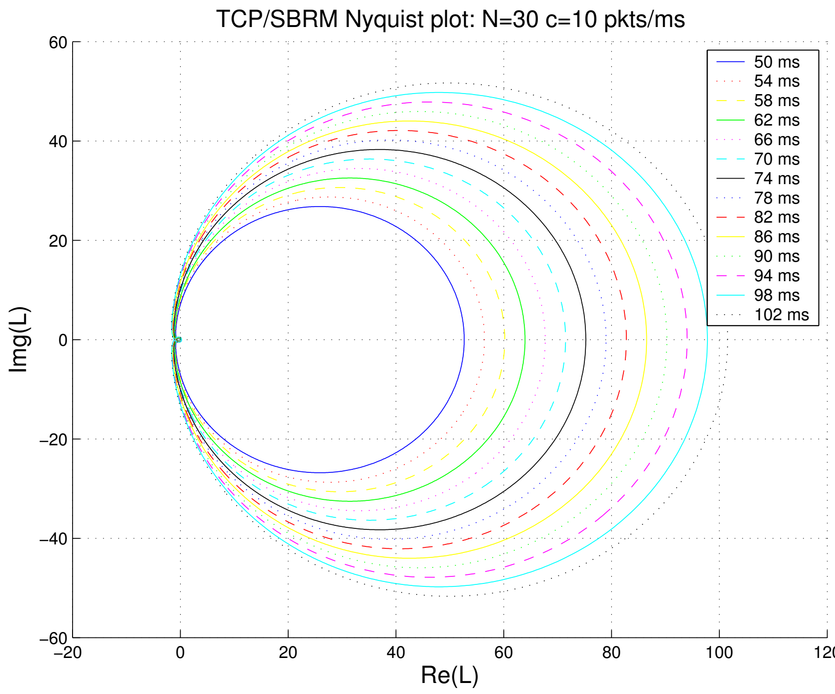
50 ms to 102 ms in steps of 4 ms. Then the round-trip time is determined at which the contour of the Nyquist plot is closest to $(-1,0)$. This is roughly the round-trip delay at which the system becomes unstable: If the round-trip delay is larger than this value, the system will oscillate (cf. Section 6.3). This point will be referred to as *critical delay* d_{crit}.

Figure 6.6.1 shows examples for SBRM, and Figure 6.6.2 shows examples for TCP Reno with RED queues. The RED model used here was taken from [LPW$^+$02], in which a similar derivation of a control theoretic model was published for TCP Reno with RED queues. This model is used to benchmark the proposed SBRM approach. Additionally, the simulations used for validation of the model in [LPW$^+$02] were re-conducted.

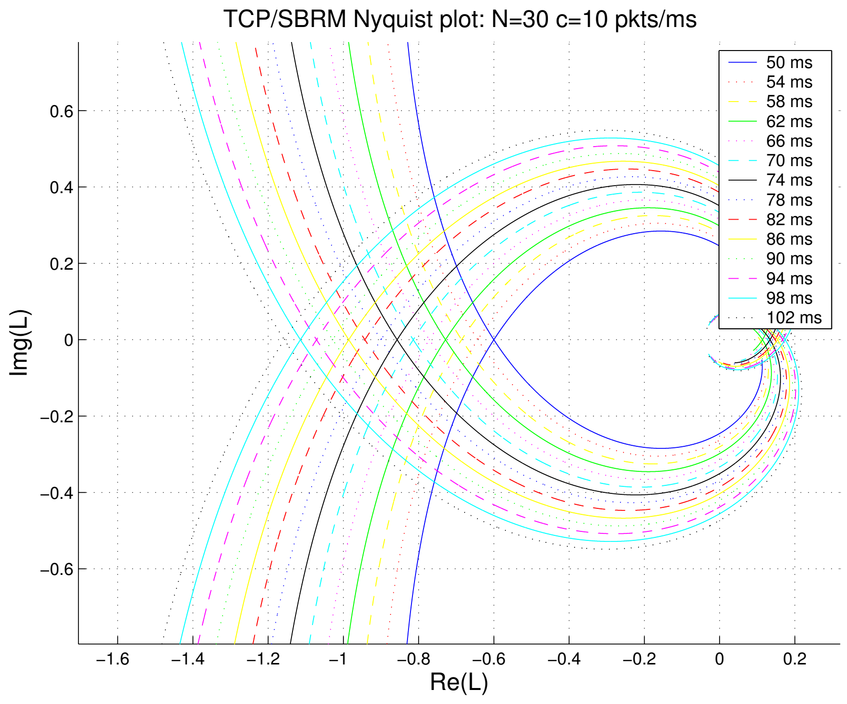
Further, the frequency is determined at which the Nyquist contour crosses the x–axis closest to $(-1,0)$. This is also the frequency at which the phase becomes $-180°$ in the Bode plots (cf. Figure 6.6.3), and thus the frequency at which the system will oscillate. This frequency will be referred to as the *critical frequency* f_{crit}. The well known property that the constant gain at $f_0 = 0$ strongly depends on the round-trip time for TCP Reno, is also visible in the Bode plots.

As can be derived from Figure 6.6.1b, for SBRM this critical delay is 86 ms, and for TCP Reno+RED the critical delay is 62 ms (cf. Figure 6.6.2b). Table 6.6.2 summarizes the critical delays and frequencies for a different number of sources and link capacities.

The results show the tendency that the critical round-trip time decreases as the capacity of the link increases. This is a significant finding for actual networks. While in the past the backbone network was mostly the limiting factor, nowadays the access link is often the bottleneck. The readily available capacity of the access link has increased in the last 15 years from 2400 bps to more than 8 Mbps. In the future, even higher access link capacities will be available. The round-trip delay, on the other hand, cannot be decreased arbitrarily. It consists of the components transmission delay, packet assembling delay and queuing delay; only the queuing delay will decrease as the capacity increases. Thus, from these findings, instability will be become a

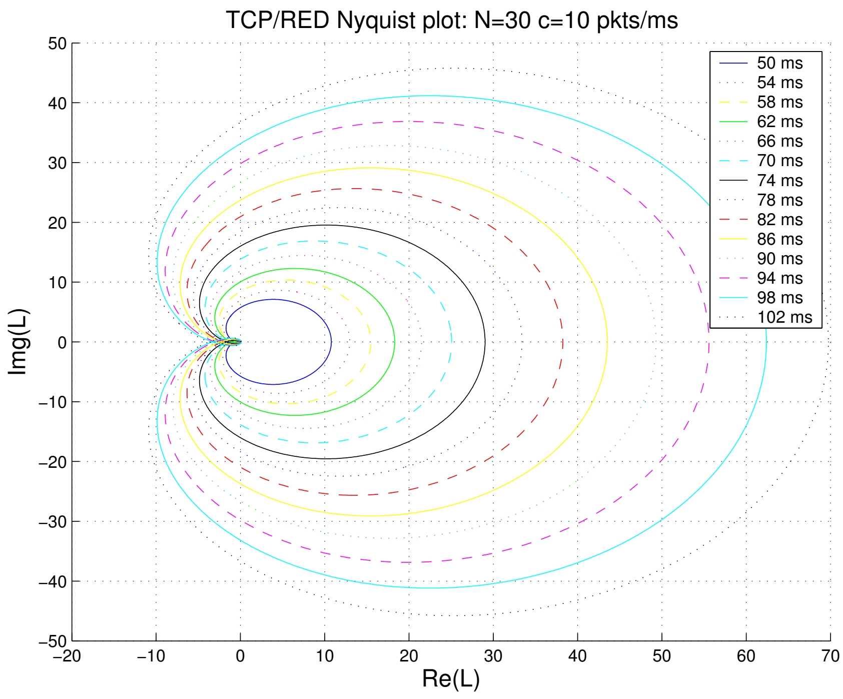


(a) SBRM

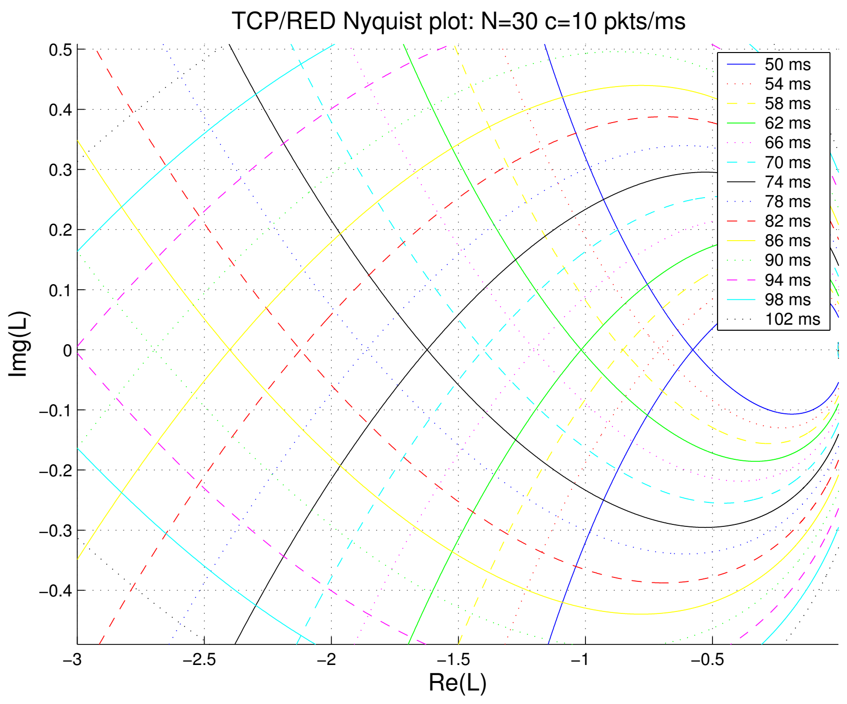


(b) SBRM (zoom)

Figure 6.6.1: Nyquist plots: Determination of critical delay (SBRM, N=30, c=10)

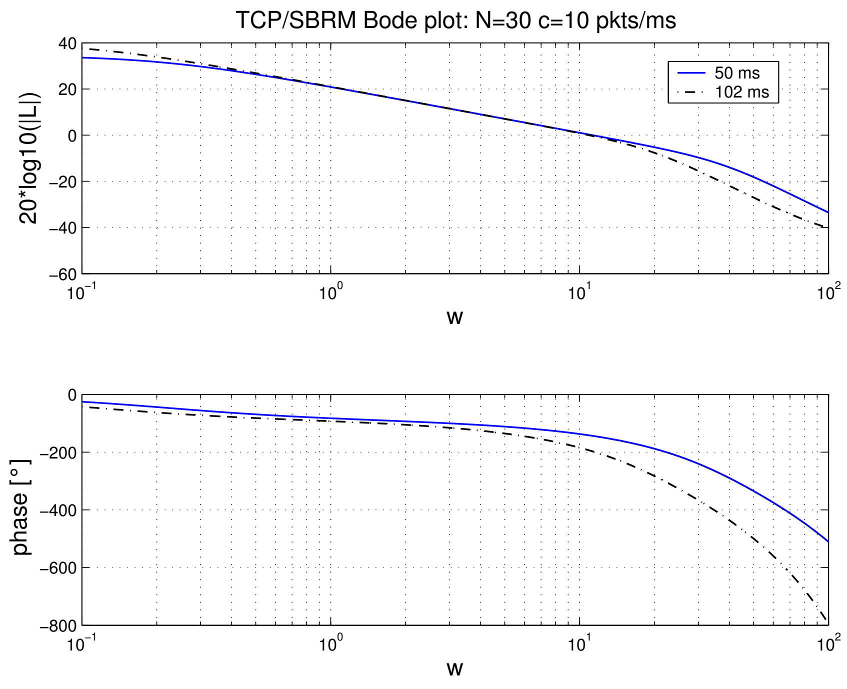


(a) TCP/Reno+RED

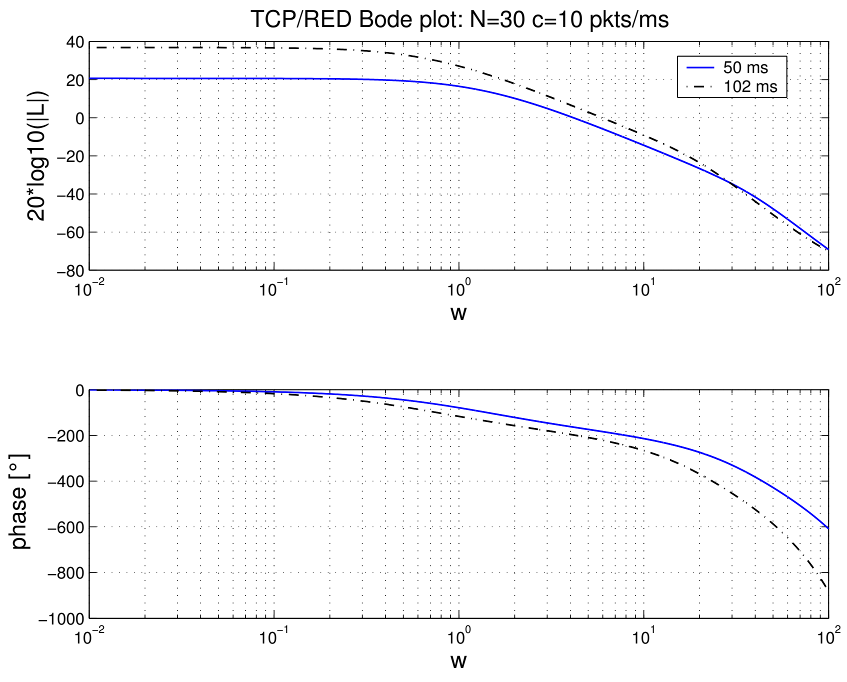


(b) TCP/Reno+RED (zoom)

Figure 6.6.2: Nyquist plots: Determination of critical delay (TCP Reno+RED, N=30, c=10)



(a) SBRM



(b) TCP/Reno+RED

Figure 6.6.3: Bode plots: Determination of critical delay

Table 6.6.2: Critical delays and critical frequencies of TCP Reno+RED and SBRM

N	c [pkts/ms]	d_{crit} [ms] Reno+RED	d_{crit} [ms] SBRM	f_{crit} [Hz] Reno+RED	f_{crit} [Hz] SBRM
20	8	62	102	0.66	1.50
20	9	54	94	0.75	1.63
20	10	50	86	0.82	1.78
20	11	$\ll 50$	78	n/a	1.97
20	12	$\ll 50$	70	n/a	2.19
20	13	$\ll 50$	66	n/a	2.32
20	14	$\ll 50$	62	n/a	2.48
20	15	$\ll 50$	58	n/a	2.65
30	8	78	$\gg 102$	0.59	n/a
30	9	70	98	0.65	1.56
30	10	62	86	0.74	1.78
30	11	54	78	0.84	1.96
30	12	50	74	0.90	2.07
30	13	50	66	0.92	2.32
30	14	$\ll 50$	62	n/a	2.47
30	15	$\ll 50$	58	n/a	2.65
40	8	90	$\gg 102$	0.55	n/a
40	9	78	102	0.63	1.50
40	10	70	90	0.70	1.70
40	11	66	82	0.75	1.87
40	12	58	74	0.85	2.07
40	13	54	66	0.91	2.19
40	14	50	62	0.98	2.47
40	15	$\ll 50$	58	n/a	2.64
50	8	102	$\gg 102$	0.51	n/a
50	9	90	102	0.58	1.50
50	10	82	90	0.64	1.70
50	11	74	82	0.70	1.87
50	12	66	74	0.79	2.07
50	13	62	70	0.84	2.19
50	14	58	62	0.90	2.47
50	15	$\ll 50$	58	n/a	2.64
60	8	$\gg 102$	$\gg 102$	n/a	n/a
60	9	98	102	0.56	1.50
60	10	90	94	0.61	1.63
60	11	82	82	0.67	1.87
60	12	74	74	0.74	2.07
60	13	66	70	0.82	2.19
60	14	62	66	0.88	2.33
60	15	58	58	0.94	2.64

significant problem. For example, typical round-trip delays on the Internet are between 20 ms (national connections) and 150 ms (international connections). Assuming a bottleneck capacity of 100 Mbps, RED queues, packet sizes of 1500 bytes, and 50 active sources, the system will be unstable for round-trip delays roughly larger than 100 ms. Also, for TCP Reno with RED, the critical delay strongly depends on the number of active sources. If only 20 sources are active, instability will already occur when the round-trip exceeds 60 ms. For SBRM, there is only a slight dependency on the number of sources.

Figure 6.6.4 shows the stability regions that can be derived from the analysis. The stable region is in the lower left corner. As the capacity of the link increases, the round-trip delay must decrease to maintain stability. The figure also shows clearly that for TCP Reno+RED there is a strong dependency on the number of sources. The fewer the number of sources, the smaller the stability region. This dependency is significantly reduced for SBRM.

Generally, control theoretic models can also be used to calculate the parameters of the RED or SBRM algorithm to ensure stability for the environment in which it is being deployed. For example, stability should be ensured for all round-trip delays that are typical on the Internet. A network operator can thus utilize the model to calculate queue parameters such that the stability region is large enough for the network. Such a model could also be used to change the source algorithm altogether as was done for the FAST [FAS03] proposal.

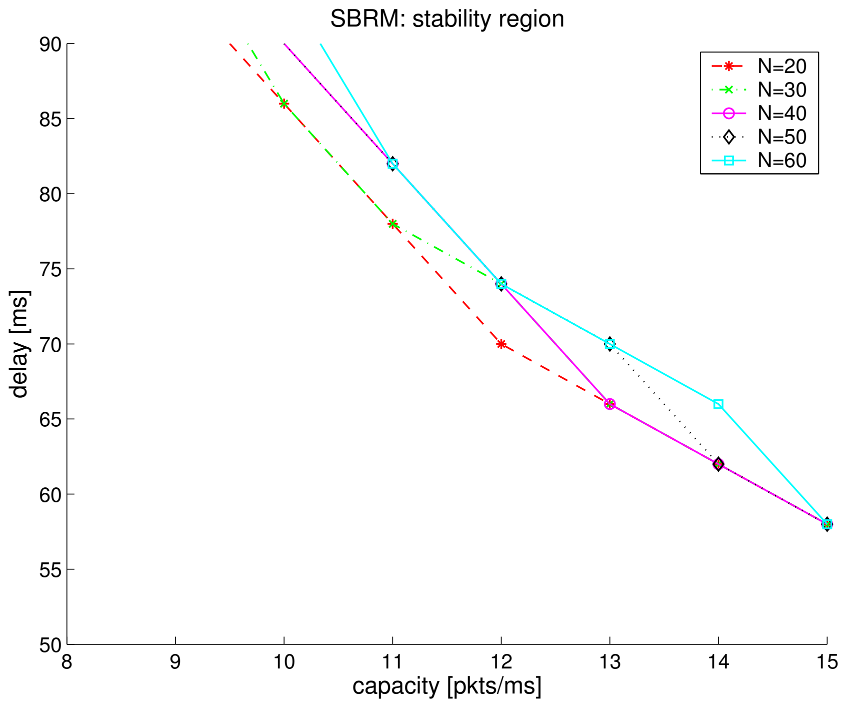
6.6.2 Recommendations for SBRM

To increase stability without completely changing the algorithms, the parameters should be chosen such that the gain margin or phase margin becomes greater (cf. Figure 6.3.4). In SBRM, four parameters can be modified: $\overline{\kappa}$, w, γ, and b_0. The effect of changes to these parameters can be examined using the transfer function model given by (6.5.21). Figures 6.6.5 and 6.6.6 show the resulting Bode plots for the variation of these parameters.

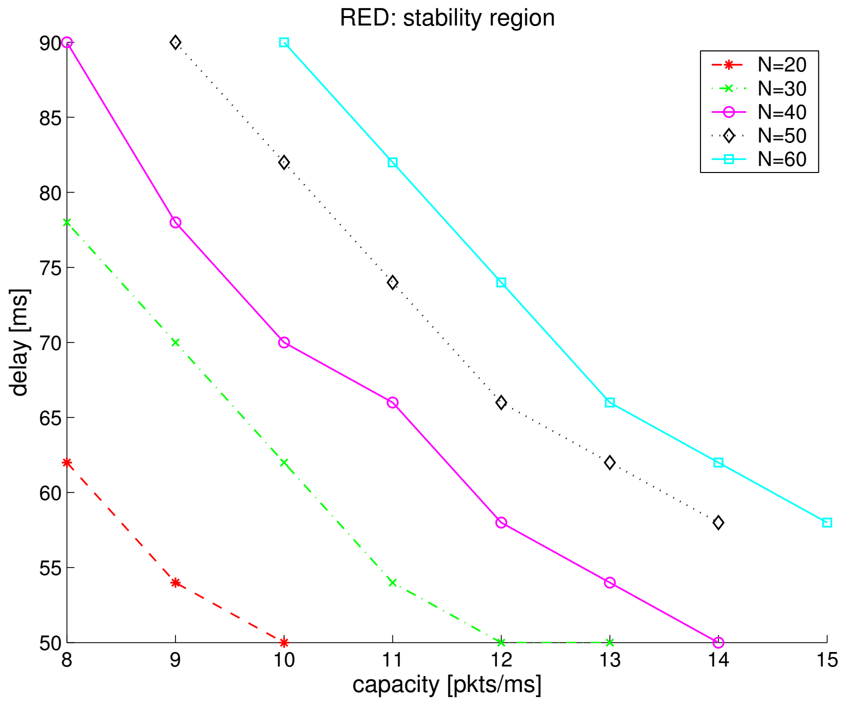
To improve stability, $\overline{\kappa}$ should be as small as possible. On the other hand, this also has an impact on the convergence speed, which should be high enough to react to changing network conditions. As already discussed in Section 6.2, good stability is not the primary goal, congestion avoidance and fast reaction to changing network conditions are more important. Therefore the author of this dissertation suggests choosing a larger $\overline{\kappa}$ for new connections, then reducing it after a few round-trip times (also cf. Subsection 5.6.4). $cwnd^*$ also increases the gain in (6.5.21), thus $\overline{\kappa}$ could also dynamically be adjusted to compensate the gain increase by $cwnd^*$. Both dynamic adjustments can be implemented in the source without the need for modification in the rest of the network. The impact of the *willingness to pay* on convergence is minimal, it should be chosen according to bandwidth preferences.

A network operator, on the other hand, can select the values for b_0 and γ. The choice of b_0 has little impact on stability: The larger the b_0, the larger the gain margin is. However, it is more important to choose b_0 to optimize utilization and average queue length. Note that the control theoretic model was based on the assumption that the queue does not run empty. Therefore b_0 should be chosen large enough to avoid completely empty queues when greedy sources are attached. The parameter γ should be small for optimal stability. A practical value is the inverse of the capacity of the link, $\gamma = \frac{1}{c}$.

The optimal choice of all parameters depends on the number of sources sharing the single bottleneck link. While this can be used to analytically derive limits for stability, in practice it

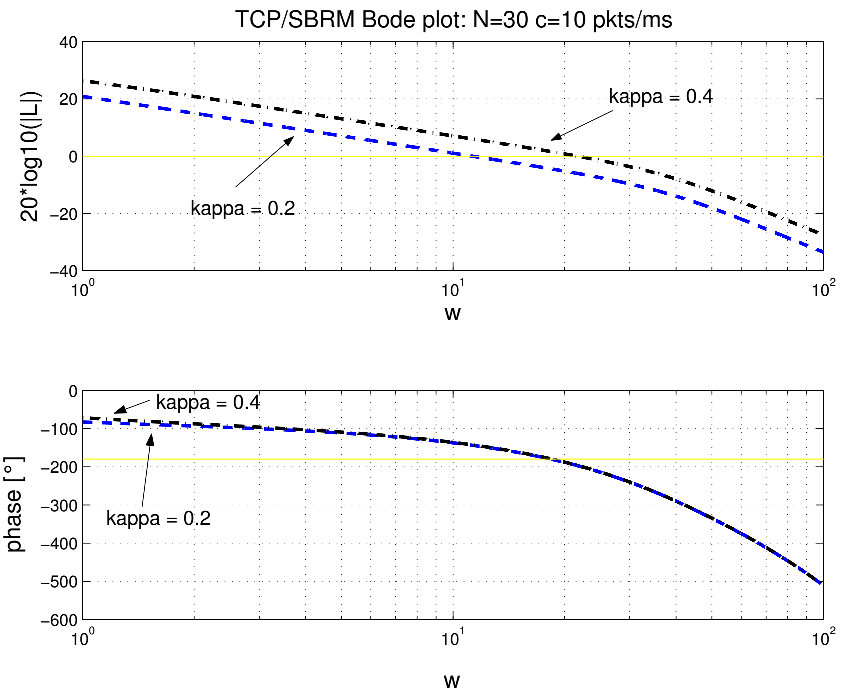


(a) SBRM

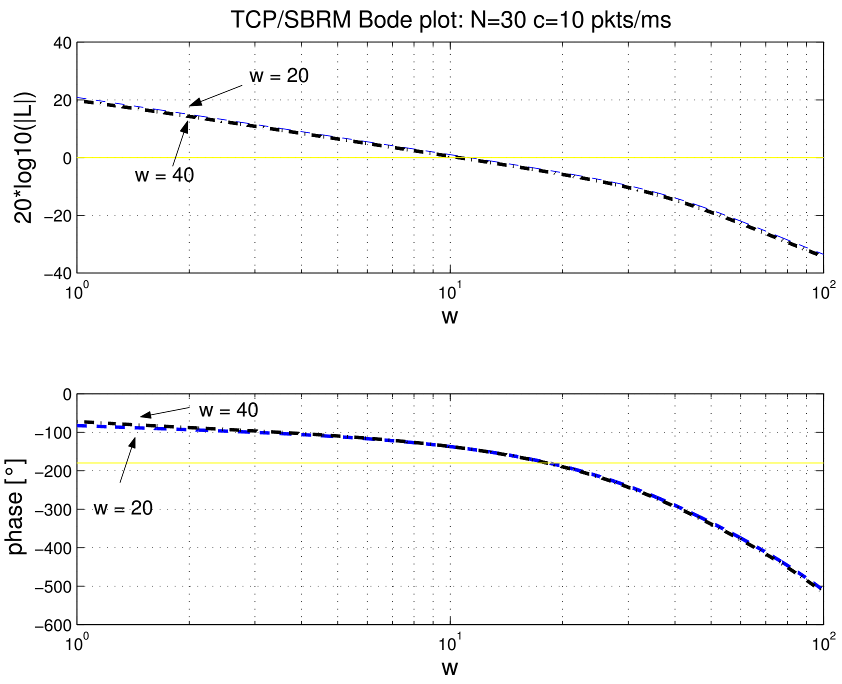


(b) TCP/Reno+RED

Figure 6.6.4: Stability regions

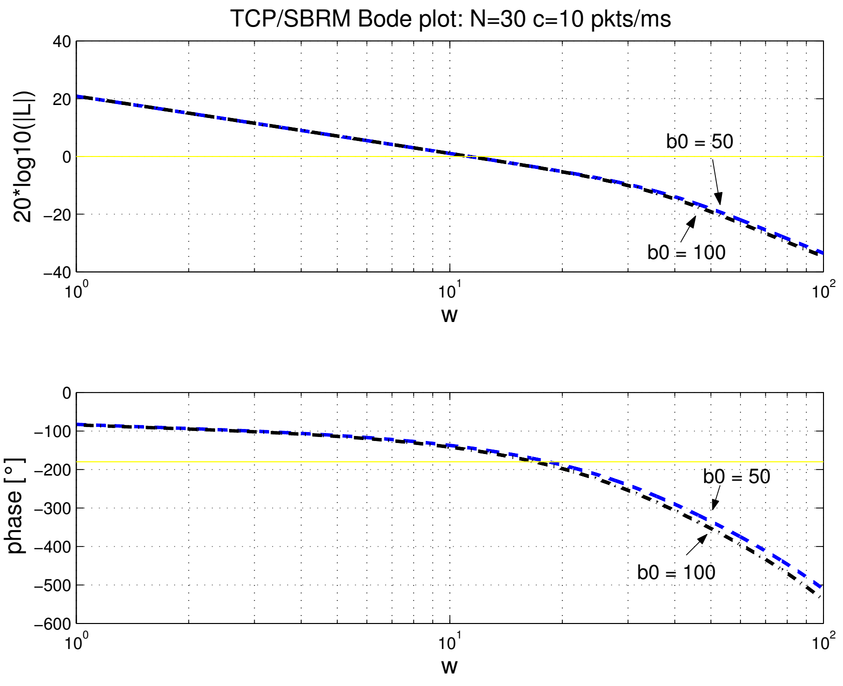


(a) Variation of kappa

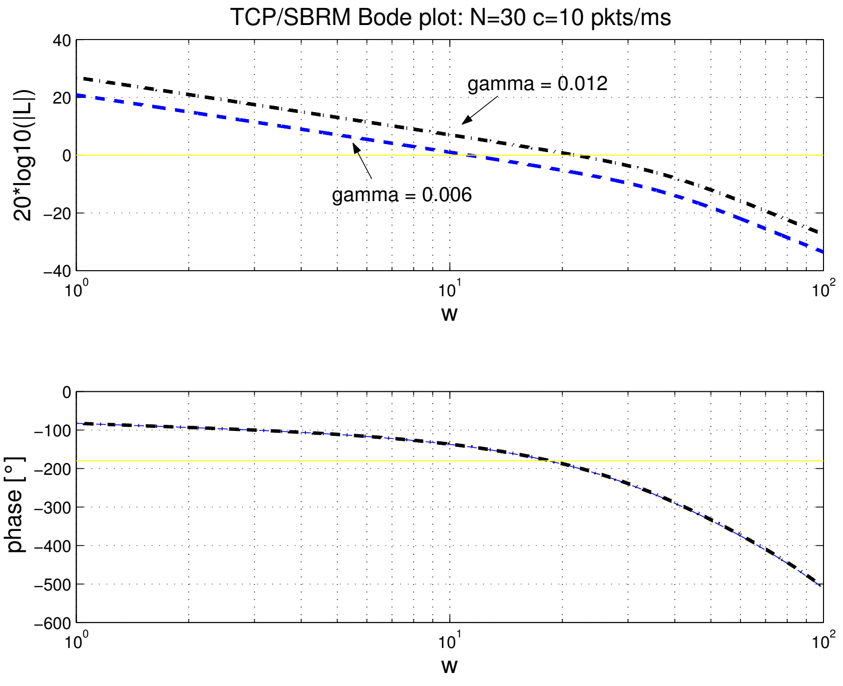


(b) Variation of the willingness-to-pay

Figure 6.6.5: Variation of SBRM parameters and influence on gain/phase margins (κ and w)



(a) Variation of b0



(b) Variation of gamma

Figure 6.6.6: Variation of SBRM parameters and influence on gain/phase margins (b_0 and γ)

is not feasible as the number of sources using a bottleneck link is not known. However, in the queue it could be possible to build a hash table of source IP addresses to count the number of sources. This information could then be used to adjust γ. Since significant changes are necessary to derive implementable bounds for the parameters to ensure stability, this will be left for future research.

6.6.3 Validation of the Linearized Model

To validate the results obtained in the stability analysis using the linearized model, the original non-linear model given by the differential equations (6.4.2)–(6.4.7) was simulated in *Simulink*. Again, for comparison, a model for TCP Reno+RED was also implemented, which was taken from [LPW$^+$02]. The implemented models are shown in Figure 6.6.7.

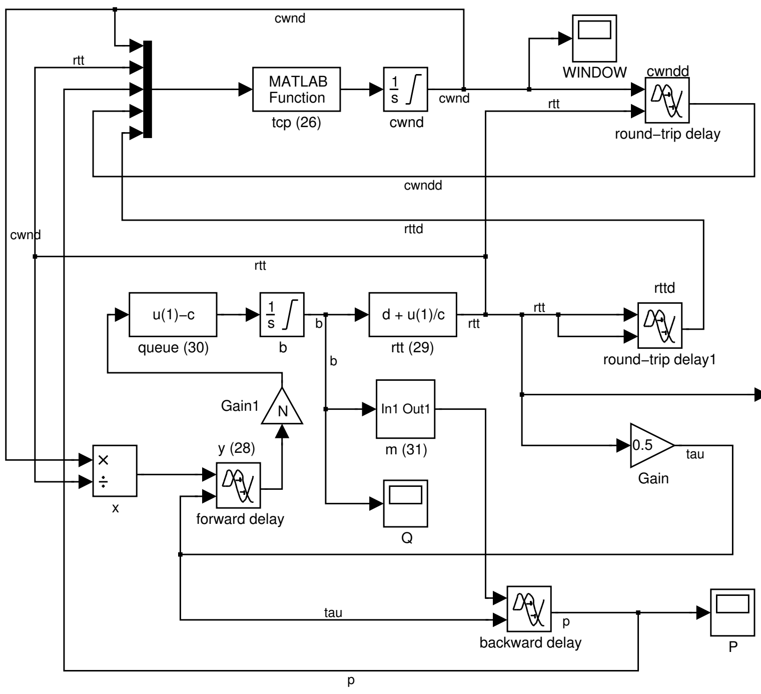
The development of the queue length over time was then recorded for the same sets of parameters used in the previous analysis. Figure 6.6.8 shows the queue size trajectory of the *Simulink* model for SBRM. The corresponding plots for TCP Reno+RED are shown in Figure 6.6.9. Up to a certain minimum round-trip delay the system is stable. If the delay becomes larger, the system will be unstable. The difference between a stable and an unstable system is clearly visible.

Using the queue plots, for each parameter set the critical delay was determined at which the system changes from stable to unstable. These values were then compared to the critical delays obtained previously from the Nyquist plot analysis. For each parameter set, the critical delays from the model (x–axis) were plotted vs. the simulation (y–axis). Good matches are scattered around the identity line (cf. Figure 6.6.10a). The frequency of the oscillations (critical frequency) was also determined for both model and simulation and plotted them (cf. Figure 6.6.10b). Considering a resolution of 4 ms, the figures show a nearly perfect match for critical delay and critical frequency. Thus, the simulation corroborates the transfer function model (6.5.21) with respect to characteristic instability parameters. Therefore, linearization is a valid approach for stability analysis. Similarly, this conclusion also holds for TCP Reno+RED, as shown by Figure 6.6.11.

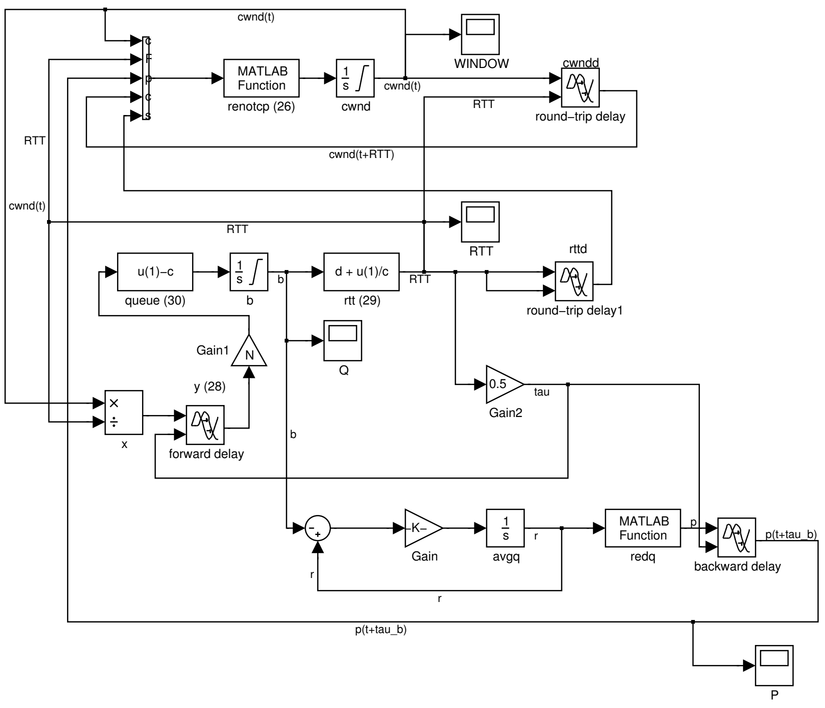
These results were also validated using actual implementations of SBRM and TCP Reno+RED in the *ns-2* network simulator [UCB]. Since these simulations are packet based, there will be differences to the continuous transfer function models. The queue size will always change when a packet arrives. Furthermore, the sources only send packets when an acknowledgment arrives. These effects will introduce additional types of oscillations to the queue.

After an initial phase to allow reaching of the steady-state, the instantaneous queue size was sampled at 20 Hz and recorded for 60 seconds. Again, the number of active sources, the link capacities and the round-trip delays were varied. Then a Fast Fourier Transform (FFT) was used to calculate the frequency spectrum of the queue trajectory. However, since sampling was done at 20 Hz, in the FFT only frequencies can be seen that are much lower than the packet arrival frequency, which ranges from 8 kHz to 15 kHz as the capacity ranges from 8 pkts/ms to 15 pkts/ms. Packet arrival frequencies of individual sources, which range from 133 Hz (60 sources, 8 pkts/ms link capacity) to 750 Hz (20 sources, 15 pkts/ms link capacity), are also filtered.

However, because of the other components introducing oscillations, it is difficult to clearly determine the critical delay, especially if the loop gain is low. To overcome this problem,

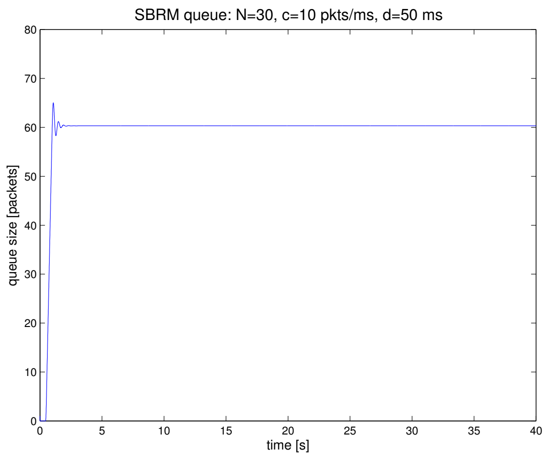


(a) SBRM

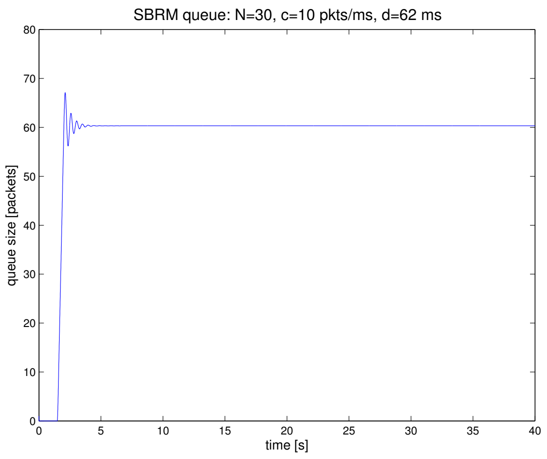


(b) TCP/Reno+RED

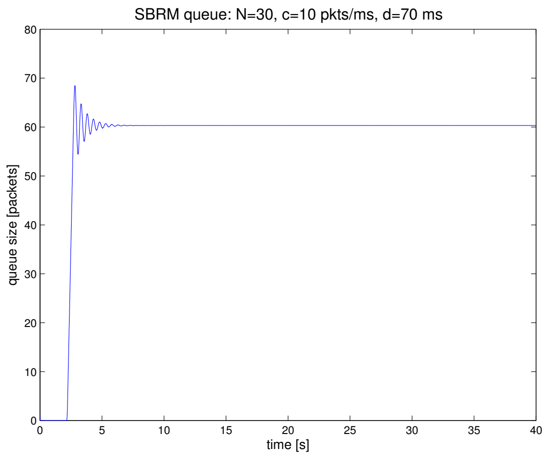
Figure 6.6.7: *Simulink* simulation models



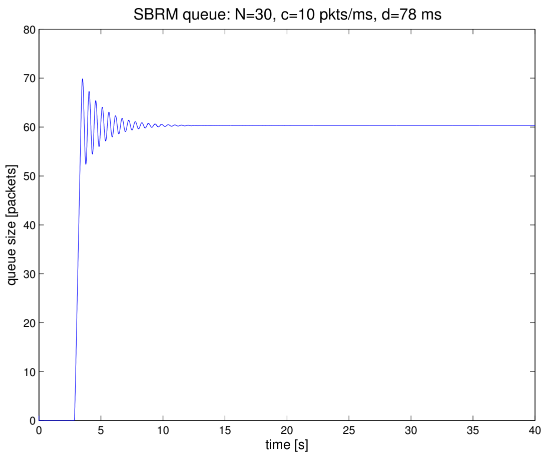
(a) 50 ms



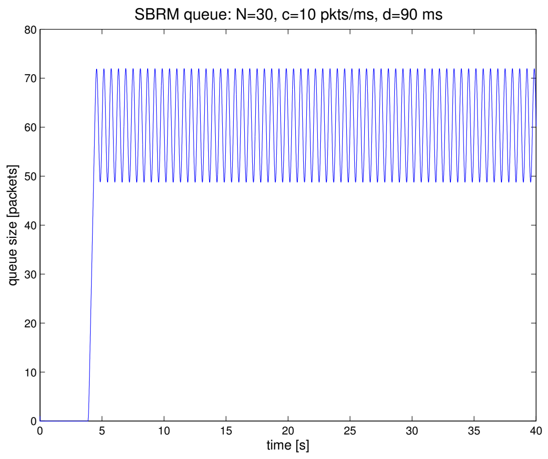
(b) 62 ms



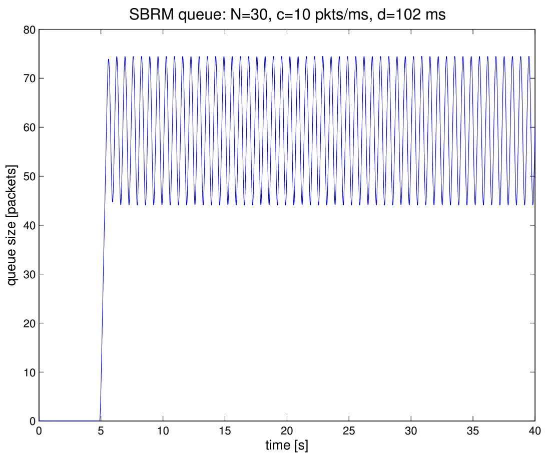
(c) 70 ms



(d) 78 ms

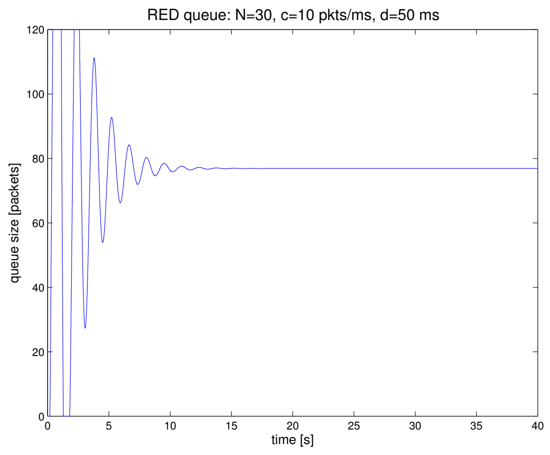


(e) 90 ms

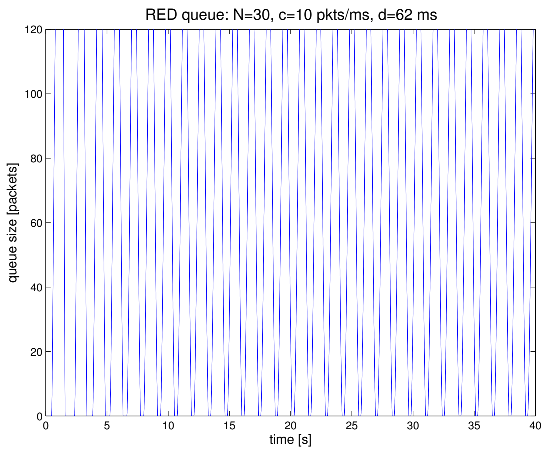


(f) 102 ms

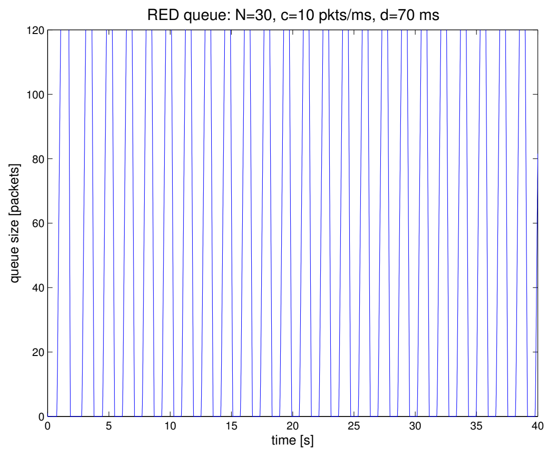
Figure 6.6.8: SBRM queue trajectory (*Simulink* simulation)



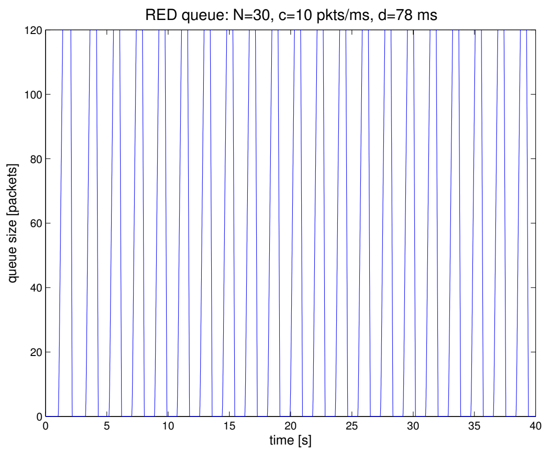
(a) 50 ms



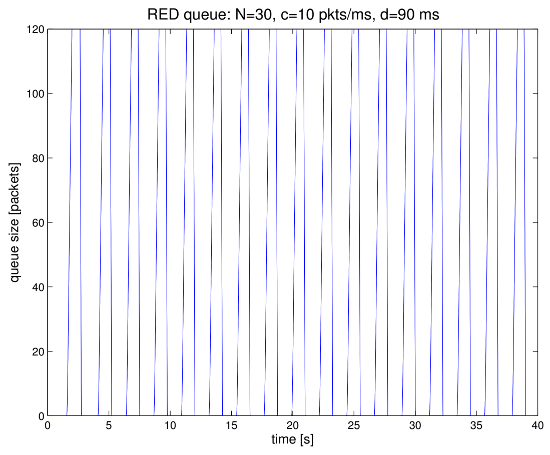
(b) 62 ms



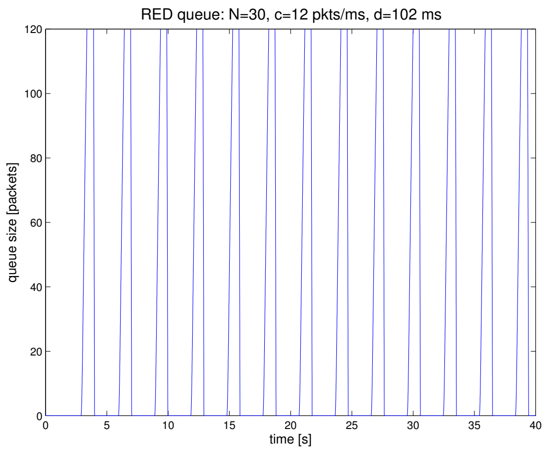
(c) 70 ms



(d) 78 ms

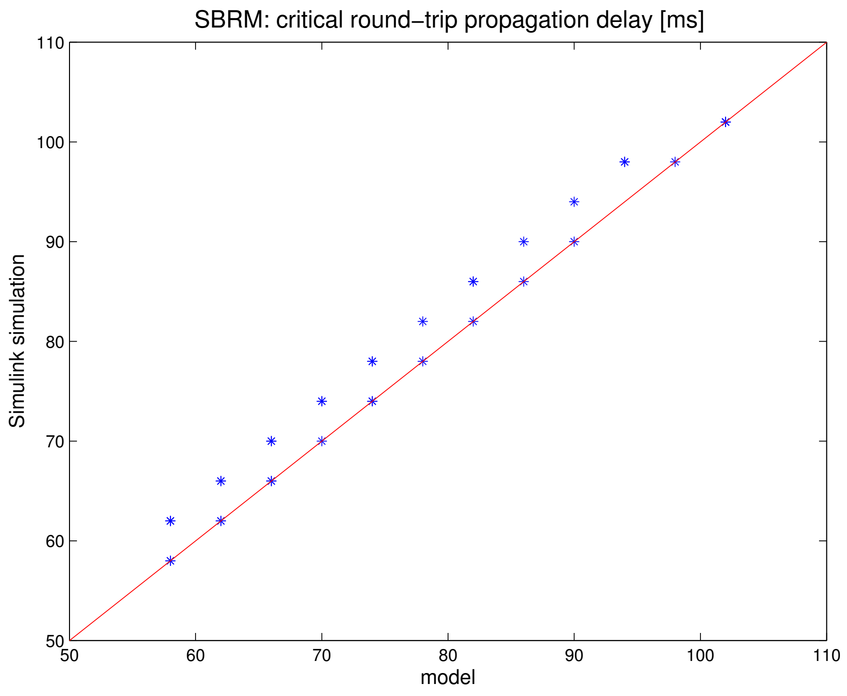


(e) 90 ms

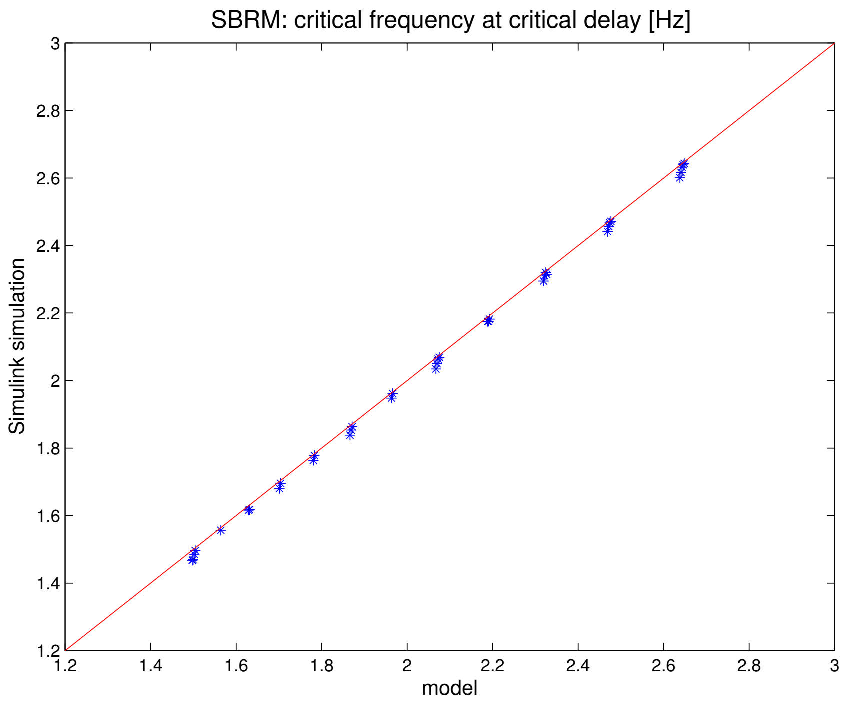


(f) 102 ms

Figure 6.6.9: TCP Reno+RED queue trajectory (*Simulink* simulation)

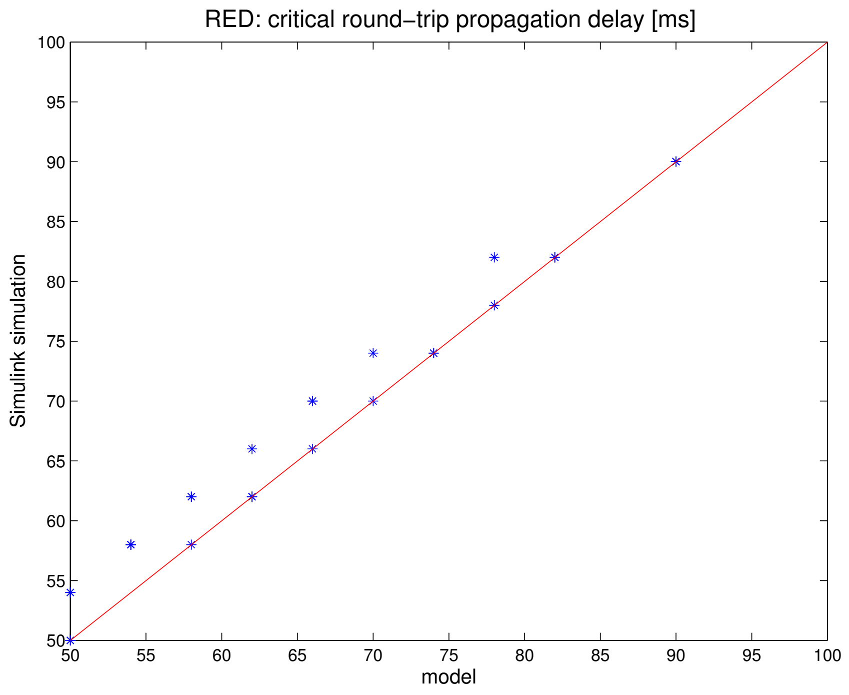


(a) Critical delays

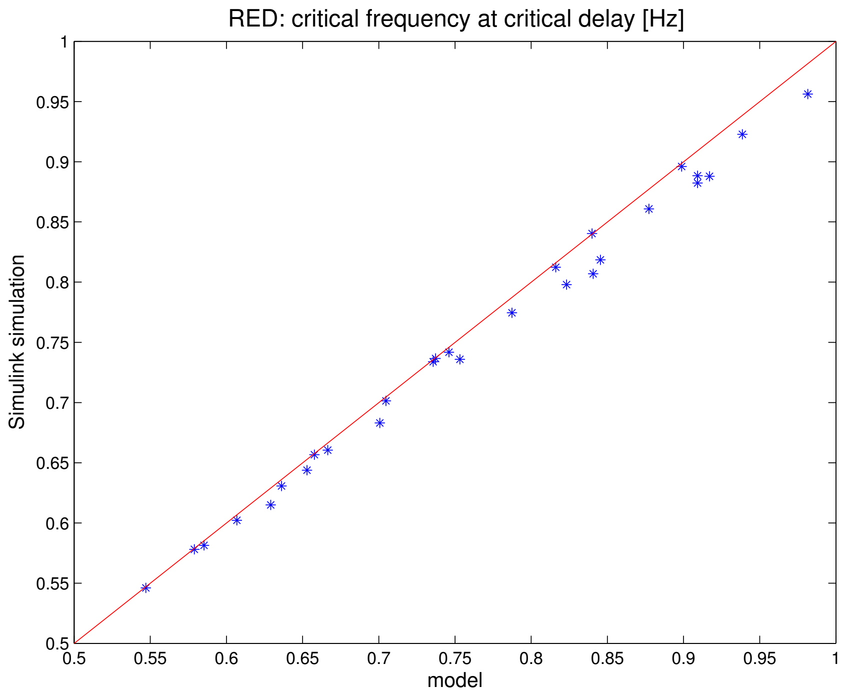


(b) Critical frequencies

Figure 6.6.10: SBRM: Critical delay and critical frequency of the linearized model and the (non-linear) *Simulink* simulation



(a) Critical delays



(b) Critical frequencies

Figure 6.6.11: TCP Reno+RED: Critical delay and critical frequency of linearized model and the (non-linear) *Simulink* simulation

only simulations at the critical delay were considered. For these simulations the Fast Fourier Transform was calculated from the instantaneous queue size. Figure 6.6.12 shows the queue trajectories and FFT plots for TCP Reno+RED. In all subfigures, oscillation due to the discrete packet arrivals are visible. In Figure 6.6.12f, however, a low frequency oscillation is obvious. This frequency is about 0.8 Hz. At this frequency, a peak in the FFT is clearly visible. To support the analysis, the MUSIC pseudospectrum was also plotted using MATLAB. The MUSIC pseudospectrum identifies sinusoidal components of a time-discrete signal [Mar87]. It shows a strong peak at the frequency identified before. The MUSIC plots were used to automatically detect this frequency by a MATLAB script.

Figure 6.6.13 shows the analogous plots for SBRM. The same process as described above was used to identify the frequencies of the oscillations. Figures 6.6.12 and 6.6.13 also show that the amplitudes of the oscillations are much higher for RED than for SBRM. This property can also be calculated for from the loop-gain models (6.5.21) and (6.5.22), or be seen in the Bode plots for both SBRM and RED (cf. Figure 6.6.3).

To validate the control models, the critical frequencies determined from the transfer function models of SBRM (6.5.21) and RED (6.5.22) were compared against the critical frequencies collected from the *ns-2* simulations. This is shown in Figure 6.6.14a for SBRM and in Figure 6.6.14b for RED respectively. Good matches are scattered around the identity line.

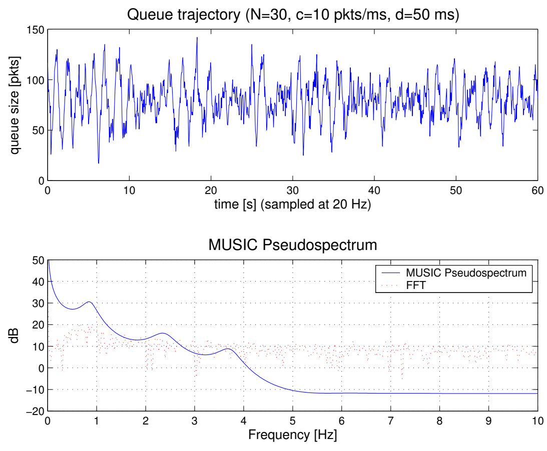
Taking into account the additional effects introduced by packet arrivals, the figures show a good match between the control theoretic models and actual implementations in *ns-2* for SBRM as well as TCP Reno+RED. Thus, validity of the control theoretic models is supported by this analysis.

6.7 Impact of Instability

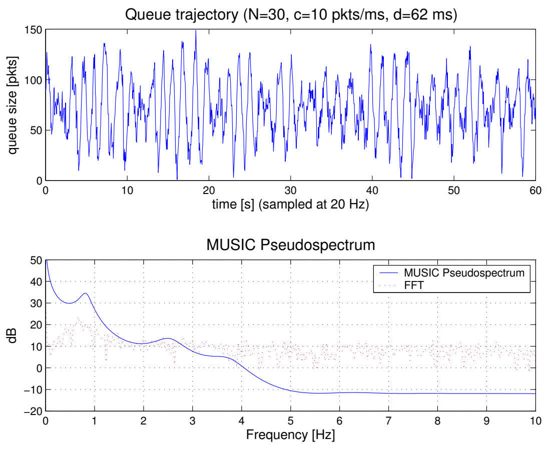
In the previous section it was shown that instability will lead to low-frequency oscillations of the queue size. In this section, the impact of instability is analyzed for actual networks. It is not only instability that impacts performance parameters: Since the instantaneous queue size will change at every packet arrival, high frequency fluctuations are naturally introduced. Further, each source will react at a frequency inverse to the round-trip delay, introducing another type of fluctuation to the queue size. For these reasons, the low-frequency oscillation due to instability might not have a direct impact on performance parameters until its amplitude becomes very large. Until then, instability might actually not be a problem at all. Therefore the impact of instability on fluid-flow model predictions of the steady-state and on performance parameters is examined.

6.7.1 Impact of Instability on Validity of Fluid-flow Model Predictions

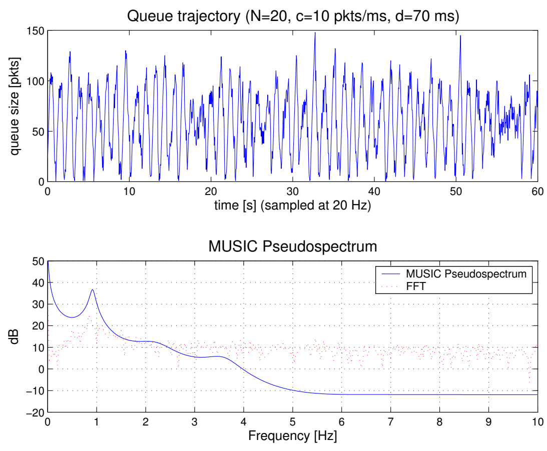
Fluid-flow models are only applicable when the system is stable. To evaluate how fluid-flow model predictions differ from the actual values when the system becomes unstable, the mean queue size and the average congestion windows were calculated for both the steady-state model and the simulation. The steady-state model is described by the the formulas shown in Table 5.5.1. Simulations of the continuous differential equation model (6.4.2)–(6.4.7) were carried out in *Simulink*. Simulations of actual implementations of SBRM and TCP Reno+RED were performed in the *ns-2* network simulator [UCB]. The same parameter sets as in the previous



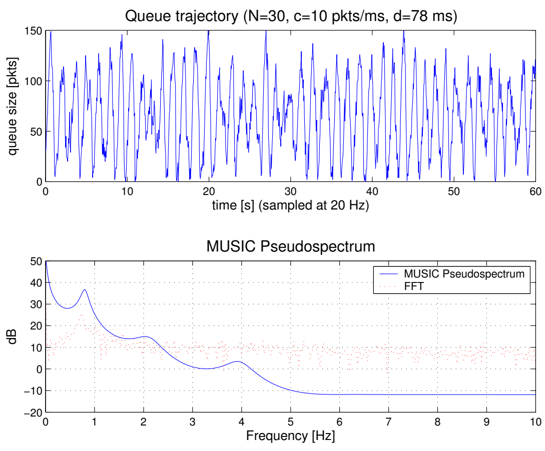
(a) 50 ms



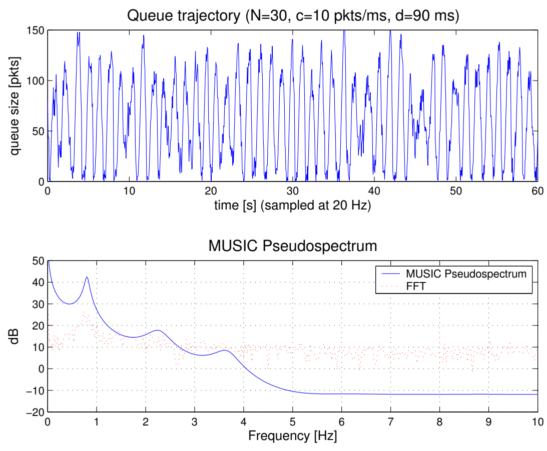
(b) 62 ms



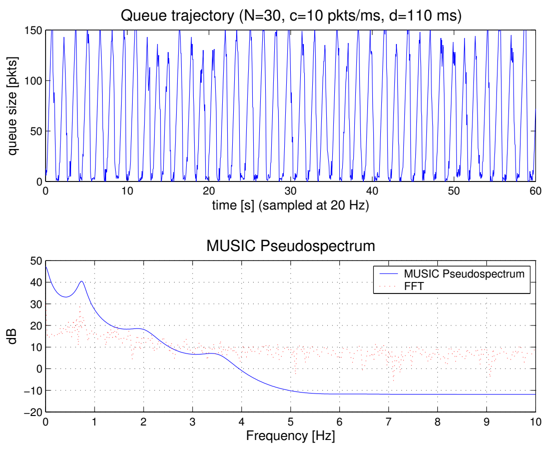
(c) 70 ms



(d) 78 ms

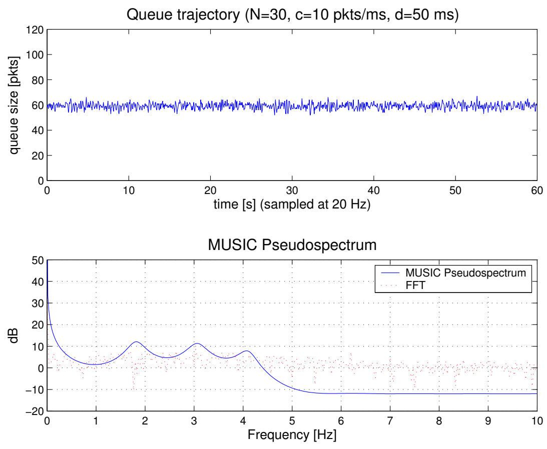


(e) 90 ms

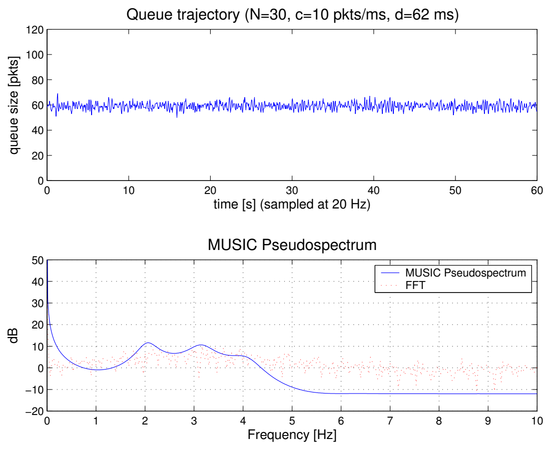


(f) 102 ms

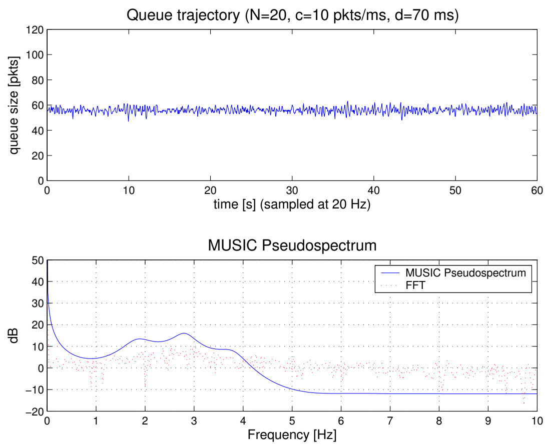
Figure 6.6.12: TCP Reno+RED queue trajectory and spectral analysis (*ns-2* simulation)



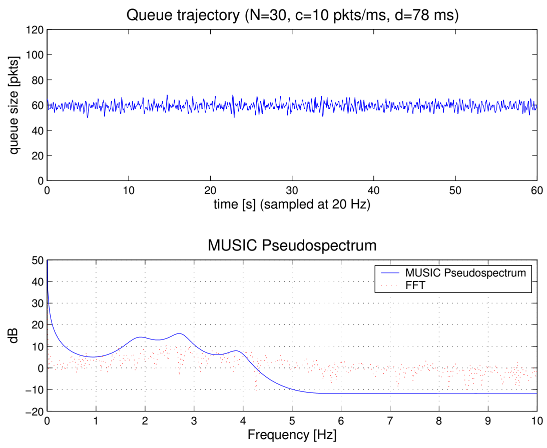
(a) 50 ms



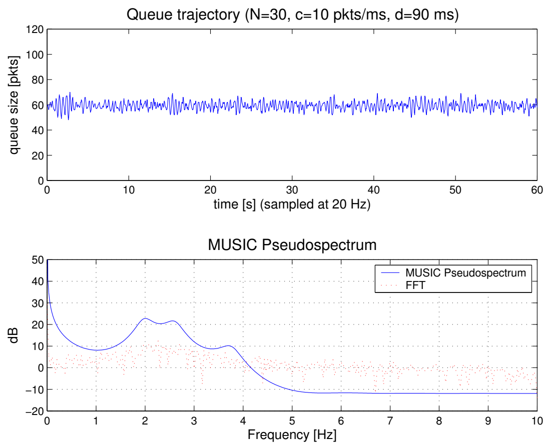
(b) 62 ms



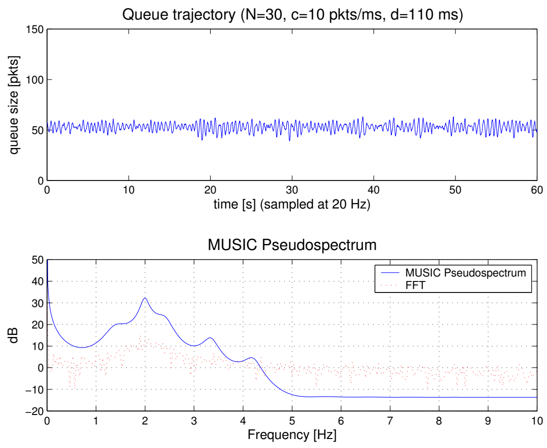
(c) 70 ms



(d) 78 ms

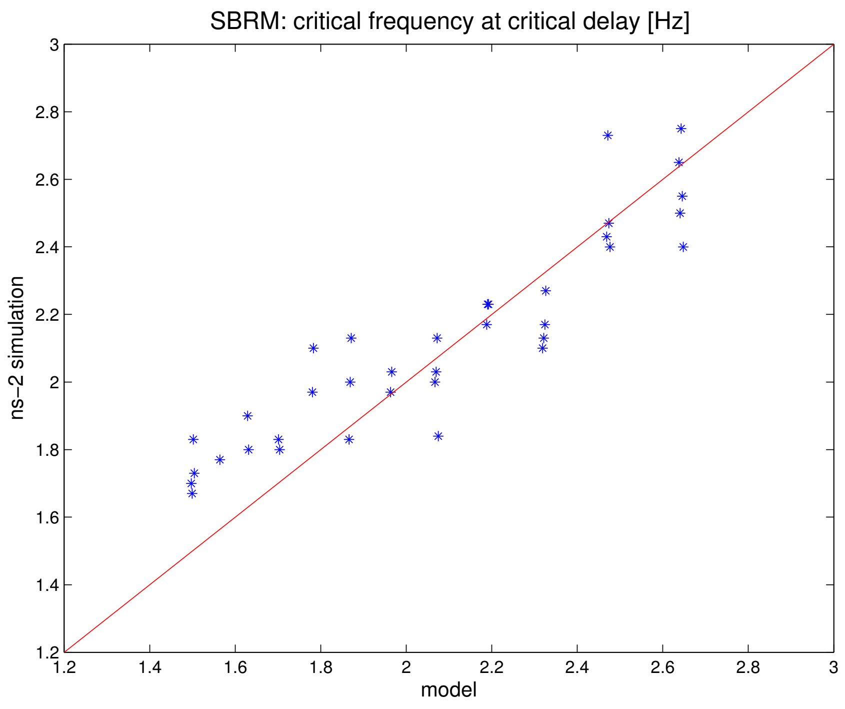


(e) 90 ms

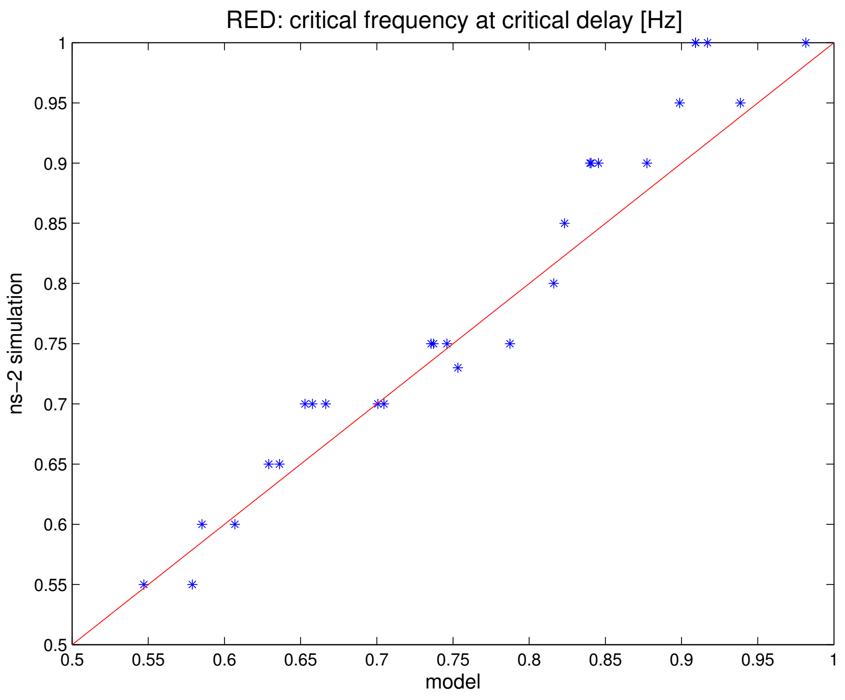


(f) 110 ms

Figure 6.6.13: SBRM queue trajectory and spectral analysis (*ns-2* simulation)



(a) SBRM



(b) TCP/Reno+RED

Figure 6.6.14: Critical frequencies of model and *ns-2* simulation

section were used. For each parameter set, the prediction for the model (x–axis) was plotted vs. the corresponding result from the simulation (y–axis). Good matches are scattered around the identity line.

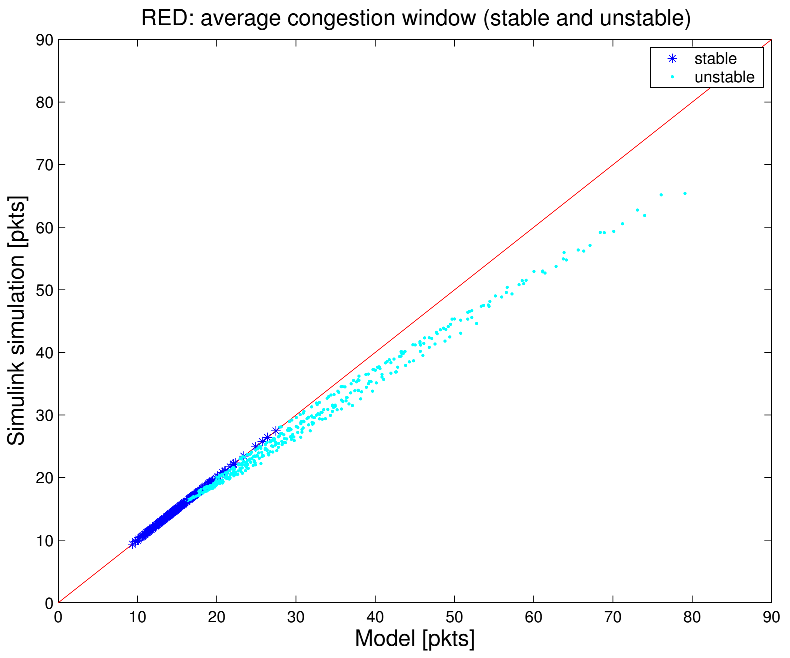
Figure 6.7.1 shows the average congestion windows and average queue sizes of the steady-state model and the *Simulink* simulations for TCP Reno with RED queues. As long as the system is stable, the match is nearly perfect (black dots). However, as soon as the system becomes unstable, differences between model and simulation are increasing (gray dots). While the mismatch between the model and simulation is only small for the average congestion window, it becomes very large for the average queue size. In both cases, the values of the simulations are smaller than the predictions made by the model. This is problematic because lower mean queue sizes further increase the chances that the queue runs empty, causing under-utilization of the link. Even worse, too low average congestion windows result in a lower sending rate than possible, also indicating under-utilization. Analogously, Figure 6.7.2 shows the average congestion windows and mean queue sizes of the steady-state model vs. the *ns-2* simulations. For the unstable region, this plot confirms the findings of Figure 6.7.1. However, for the stable region the mean queue sizes achieved by the *ns-2* simulation tend to be smaller than the ones predicted by the model. This finding is also confirmed in [Pat03]. This discrepancy is most likely a result of the queue sometimes running empty, which is not considered in the fluid-flow model [LPW$^+$02], and thus not visible in the previous plots.

The corresponding plots for SBRM are shown in Figures 6.7.3 and 6.7.4. With SBRM, the average size of the congestion window even then matches the prediction by the model when the system is unstable. Thus, average transmission rate roughly stays the same as the theoretical value — maintaining high utilization as will be shown later. However, the mean queue size becomes smaller than the prediction, similar to RED.

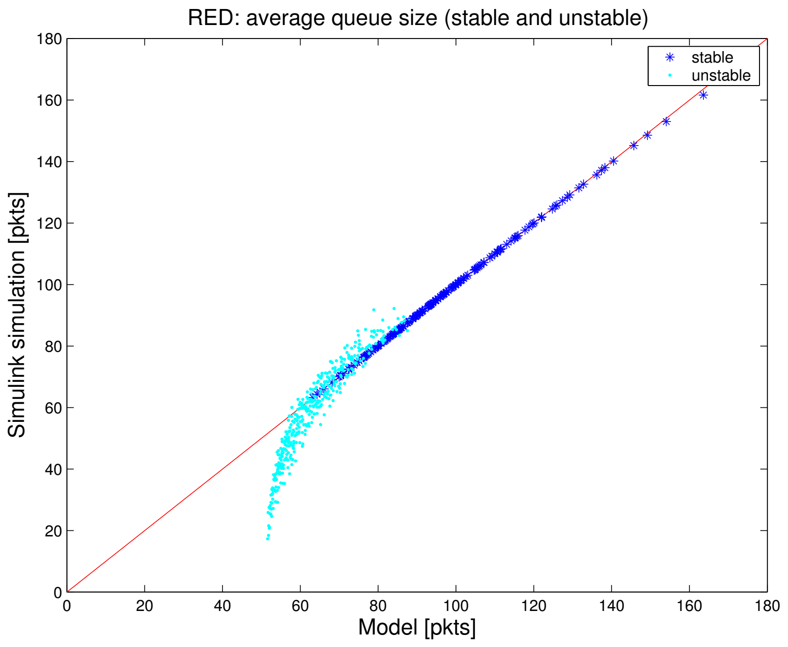
6.7.2 Impact of Instability on Performance

As mentioned before, oscillations will degrade utilization of the link: as the amplitude of the oscillation grows, the queue is more likely to run empty. Additionally, oscillations will introduce jitter which is especially bad for interactive multimedia traffic. These effects are visible in Figure 6.7.5, where mean queue size, standard deviation and link utilization derived from the 30 *ns-2* simulation runs per parameter set are shown for RED. An analysis of adaptive RED (a-RED) [FGS01] is also considered that was proposed to improve some of the previously known shortcomings of RED (cf. Figure 6.7.6). For adaptive RED, the parameters were chosen according to the suggestions made in [FGS01] (cf. Table 6.7.1). The adaptive RED algorithm is designed such that the average queue size is maintained half way between th_{min} and th_{max}. Since a mean queue size of roughly 50 packets was desired, the values 25 and 75 were chosen for these parameters. Figure 6.7.7 shows the corresponding plots for SBRM.

As previously demonstrated in Figure 6.6.3, gain at critical delay and thus standard deviation of the queue size increases as the round-trip delay increases. The mean queue size, on the other hand, decreases as the queue runs empty. This can be seen in Figures 6.7.5a, 6.7.6a, and 6.7.7a. At first, the decrease in mean queue size could be thought of as something positive, since mean queuing delay also becomes smaller. However, this is an unwanted result of the oscillations. It is only because the amplitude of the oscillations is so high that the queue runs empty, hence the mean queue size becomes smaller. At the same time the utilization of the link drops (cf. Figures

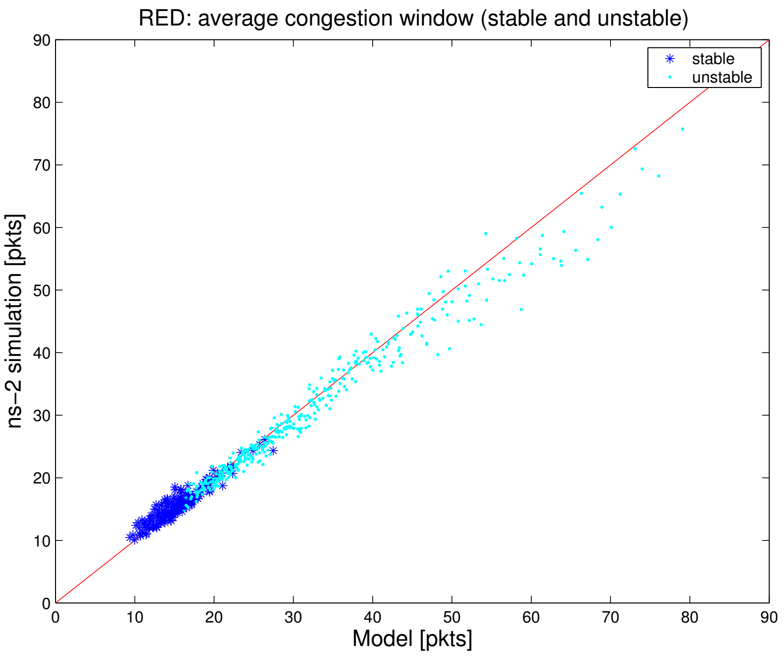


(a) Average congestion window size (model vs. Simulink)

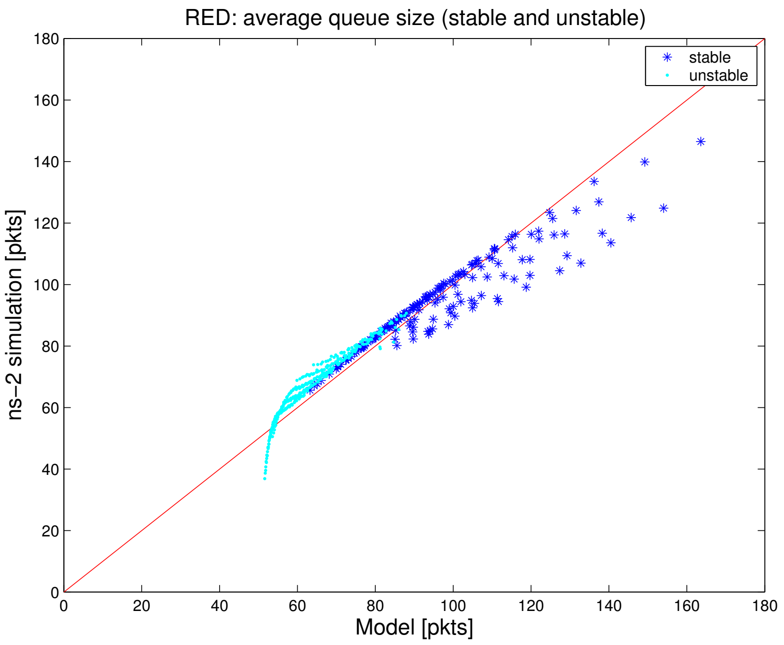


(b) Average queue size (model vs. Simulink)

Figure 6.7.1: TCP Reno+RED: Average queue and congestion window sizes of model and *Simulink* simulation

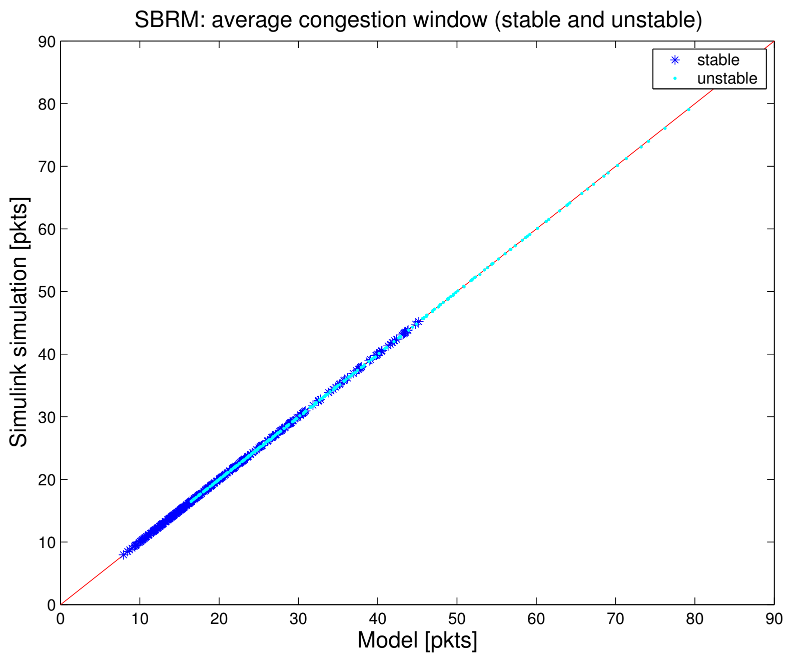


(a) Average congestion window size (model vs. ns-2)

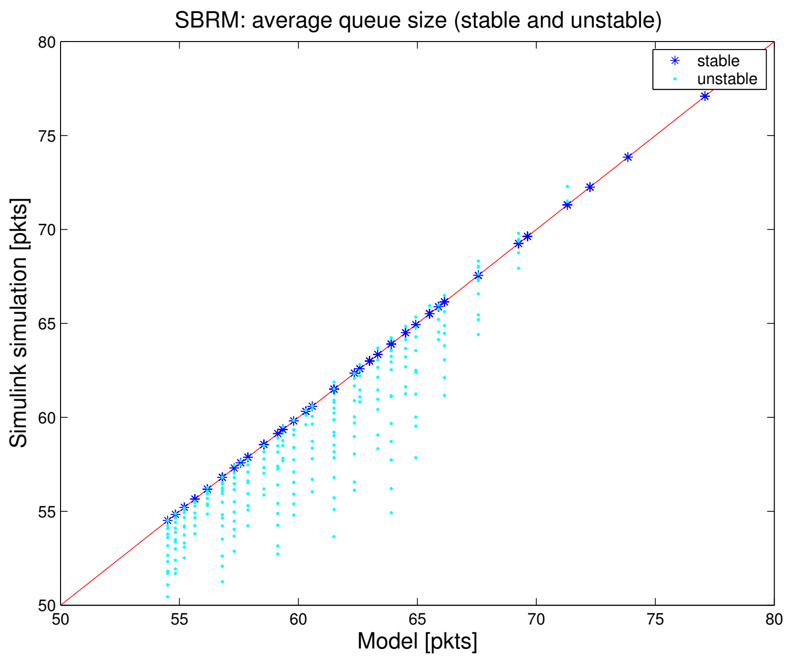


(b) Average queue size (model vs. ns-2)

Figure 6.7.2: TCP Reno+RED: Average queue and congestion window sizes of model and *ns-2* simulation

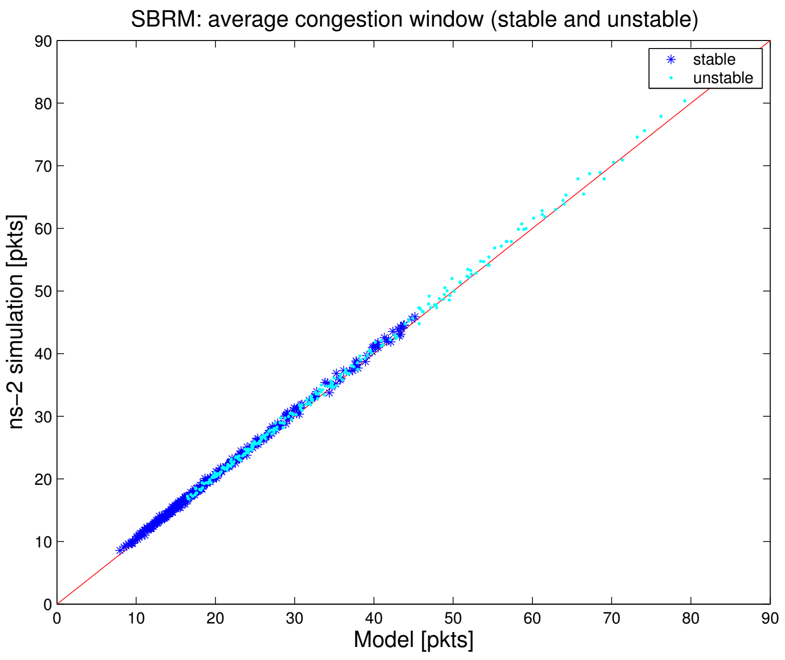


(a) Average congestion window size (model vs. Simulink)

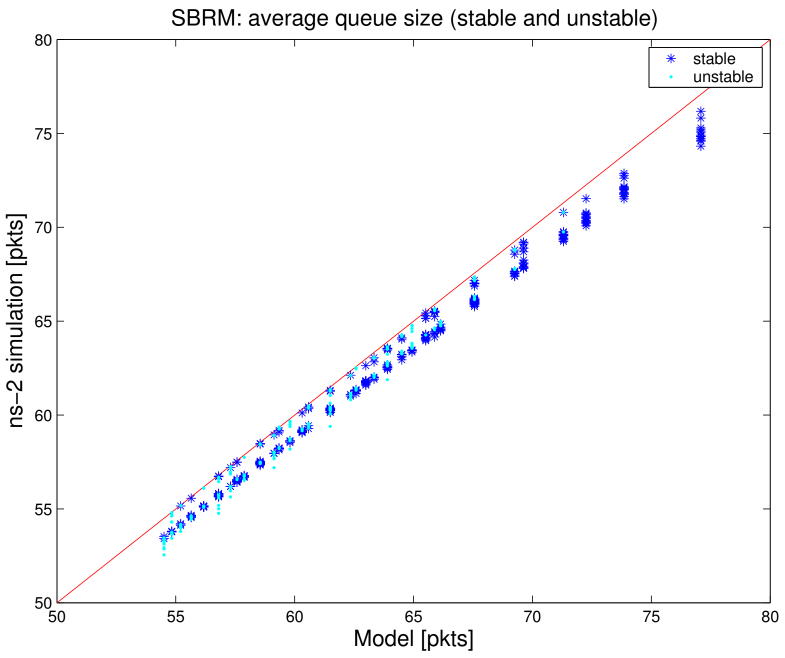


(b) Average queue size (model vs. Simulink)

Figure 6.7.3: SBRM: Average congestion window and queue sizes of model and *Simulink* simulation

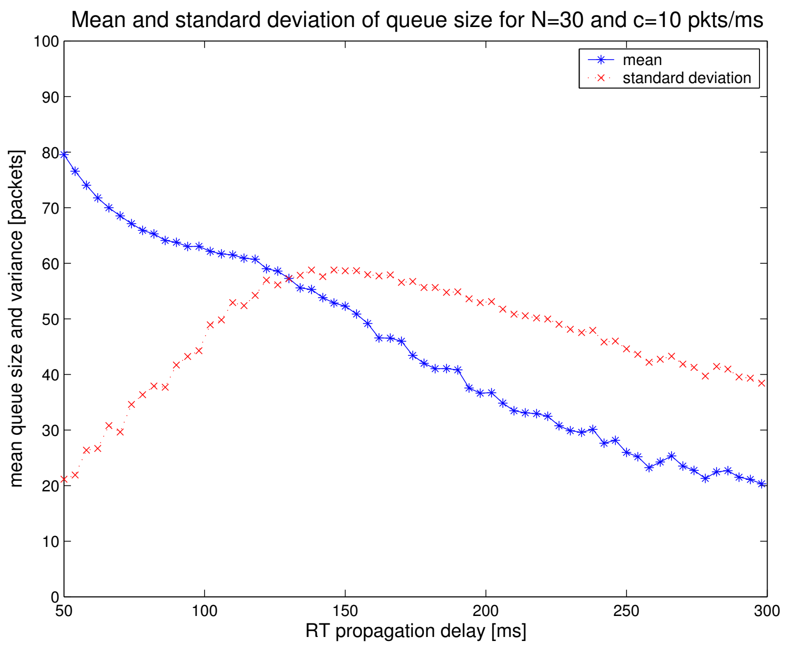


(a) Average congestion window size (model vs. ns-2)

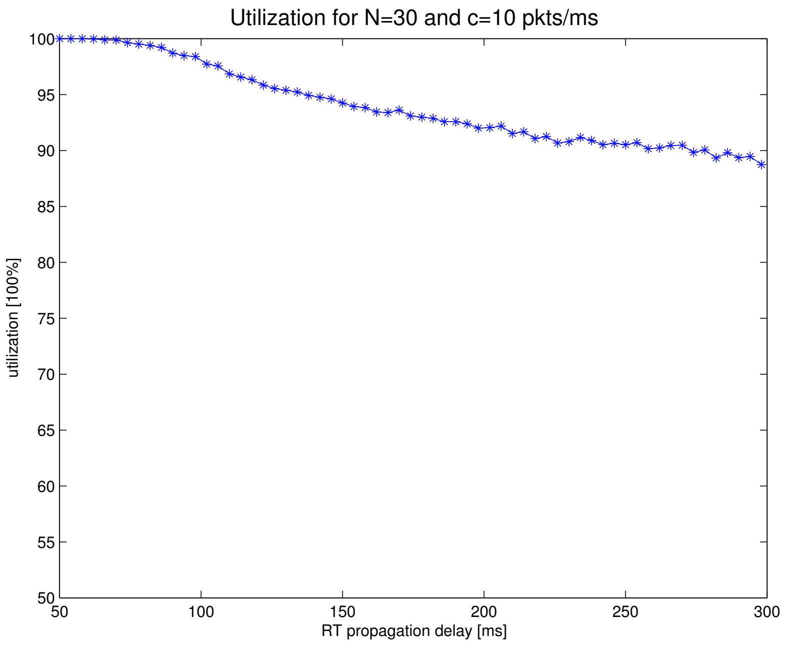


(b) Average queue size (model vs. ns-2)

Figure 6.7.4: SBRM: Average congestion window and queue sizes of model and *ns-2* simulation

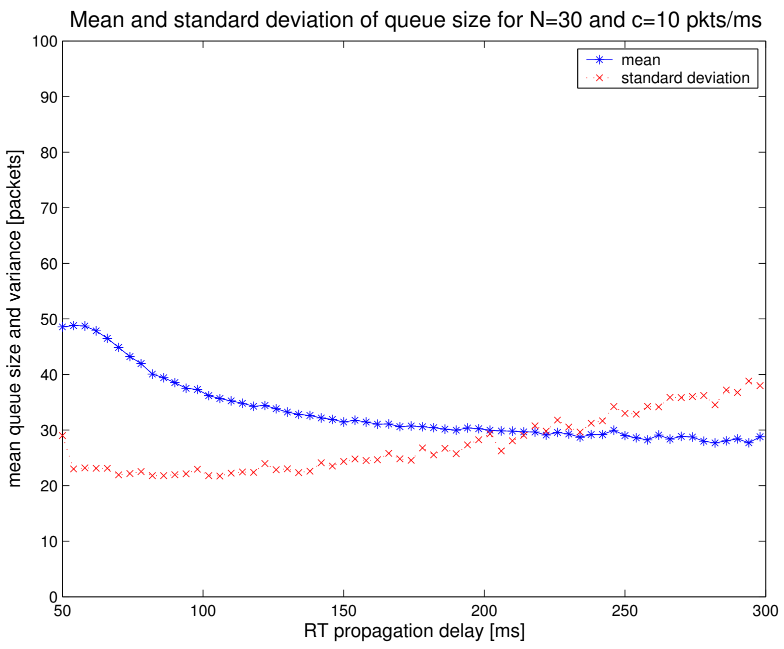


(a) Queue size

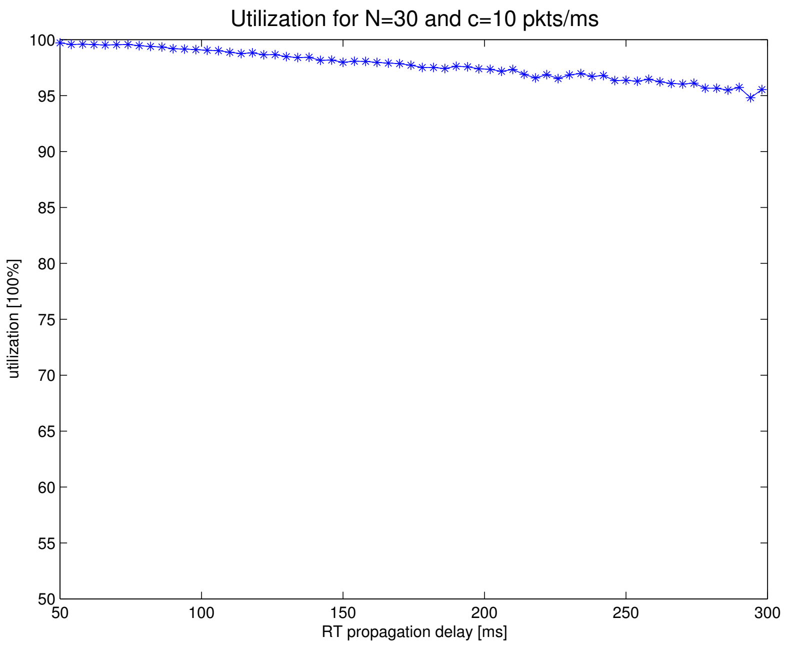


(b) Utilization

Figure 6.7.5: Bottleneck queue characteristics (RED)

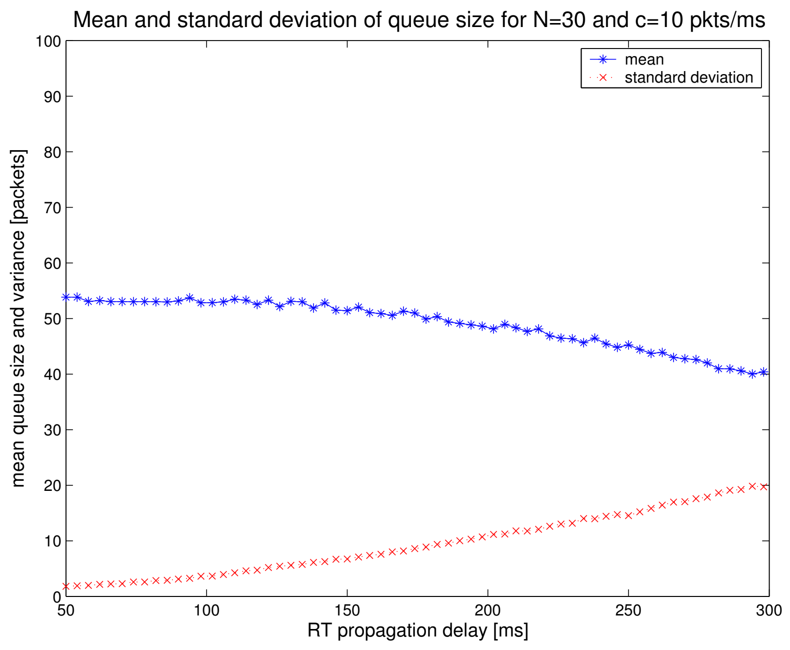


(a) Queue size

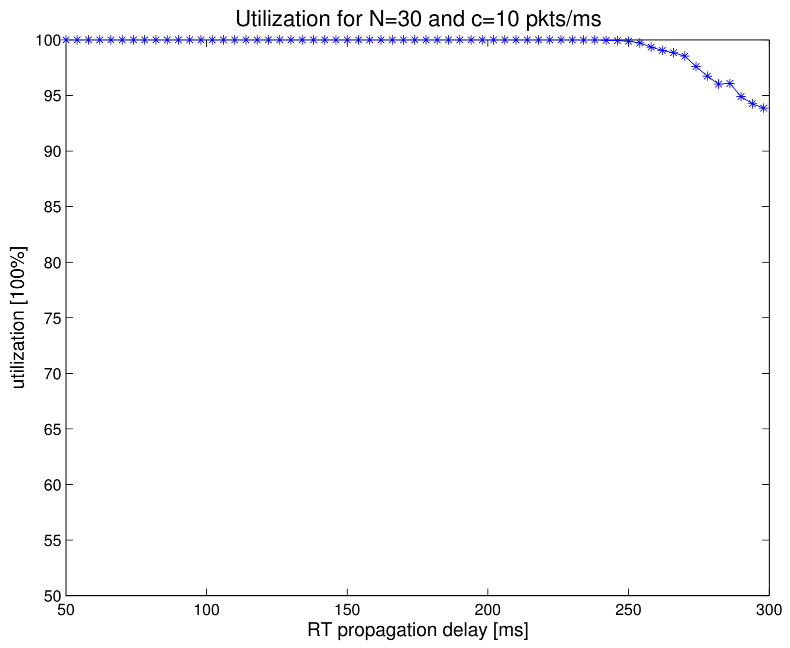


(b) Utilization

Figure 6.7.6: Bottleneck queue characteristics (a-RED)



(a) Queue size



(b) Utilization

Figure 6.7.7: Bottleneck queue characteristics (SBRM)

Table 6.7.1: Parameters used for adaptive RED simulations

TCP version	Parameter	Value
Reno/adaptive-RED	th_{min}	25
Reno/adaptive-RED	th_{max}	75
Reno/adaptive-RED	top	0.5
Reno/adaptive-RED	$bottom$	0.01
Reno/adaptive-RED	$alpha$	0.01
Reno/adaptive-RED	$beta$	0.9

6.7.5b, 6.7.6b, and 6.7.7b). Low utilization will reduce the revenue for the network operator and the overall goodput. Furthermore, at the same time as the mean queue size becomes smaller, the standard deviation becomes larger. High standard deviation is a sign for strongly fluctuating queue sizes and thus fluctuating queuing delays, which will in turn cause jitter. This is clearly something that is not wanted on a network. If the network operator did not know the control theoretic model to choose parameters enabling a stable system, he might try to increase the target queue size. This could lead to better utilization, but at the same time increase persistent queuing delay.

At some point the standard deviation becomes smaller again as round-trip time increases (cf. Figure 6.7.5a). For higher round-trip delays the queue regularly runs empty. Mean queue size is now so low such that the amplitude becomes smaller, too. Also note that the control theoretic model for RED presented in [LPW$^+$02] is valid only if the mean queue size is between minimum and maximum thresholds. Since this is not the case any more, conclusions on the amplitude of the oscillations cannot be drawn from the model.

Figure 6.7.6 shows the corresponding plots for adaptive RED (a-RED). A significant improvement is visible, utilization worsens at a much lower rate. However, again the standard deviation of the queue size is growing, indicating oscillations. Adaptive RED suffers from similar problems as RED, but the impact of instability is significantly lower than with conventional RED. SBRM, shown in Figure 6.7.7, also suffers from instability, but can maintain the target queue size and perfect utilization at a much larger range of round-trip delays than the other two variants. For the full range simulated, the amplitude of the oscillations is much smaller than with RED and a-RED. Thus, even though SBRM also is not scalable for any round-trip delay, it is stable for a much larger range and will maintain high utilization even in case of beginning instability.

6.8 Conclusions

In this chapter, a control theoretic approach was derived to evaluate scalability and stability of SBRM. The model revealed stability problems for SBRM as well, which have to be considered when choosing the correct parameters. However, the impact on performance parameters such as utilization is much lower than with RED and even lower than with adaptive RED. It was also shown that instability will lead to low frequency oscillations of the queue size and to jitter. While the resulting under-utilization of the link could partly be compensated by the network operator by setting a larger target queue size, oscillations and related jitter can only be removed by ensuring stability. Especially in the future, for networks with increasing capacities and new applications that rely on quality parameters such as delay and jitter, this will become important.

Nonetheless, as discussed initially, stability should not prevent an algorithm from reacting quickly to network congestion. Since most flows on the Internet are short-lived, stability is of lesser concern as long instability will not cause congestion. From these results it can be concluded that instead of congestion, instability will cause oscillations and thus under-utilization of the link. Taking also into account the disadvantages of congestion control algorithms that cannot only rely on binary congestion information and require full pricing information or differential delay measurements, SBRM is a good choice that performs much better than any conventional TCP variant that is being used today.

Chapter 7

Congestion Control for Inelastic Traffic

In the previous chapters, the focus was on elastic traffic that can adapt its rate according to the network's current congestion state. In this chapter, congestion control issues related to *inelastic traffic* will be discussed. Inelastic traffic is characterized by its inability to continuously gain advantage from growing available bandwidth. In the worst case, the required transmission rate is fixed and independent of available bandwidth. This property is modeled by a step utility function. If available bandwidth is below the required bandwidth, nothing can be sent and the user's utility will be zero. If required and available bandwidths match or even more bandwidth is available, only the required fixed bandwidth will be used and the user's utility is constant. On the Internet, usually media streams generate inelastic traffic.

7.1 Relevance of Congestion Control for Inelastic Traffic

Media streams commonly use the *Real-time Transport Protocol (RTP)* [SCFJ96], which in turn uses the *User Datagram Protocol (UDP)* [Pos80] for data transport. Without additional protocols, neither a *Call Admission Control (CAC)* nor closed-loop congestion control are implemented. Since UDP traffic currently makes up less than 5% of a backbone's traffic volume [Sch03], a congestion control or a Call Admission Control is generally not required. Instead, usually the user can select a stream bandwidth matching his access link rate. However, the importance of media streaming will grow and a need for protection of the network against overload is anticipated. Additionally, currently elastic traffic and inelastic traffic are not separated in the routers' queues. This could also change in the future to allow bounds on quality parameters such as delay and jitter. Without such bounds, real-time duplex streams such as Internet telephony cannot be offered at acceptable quality of service levels. Thus, congestion control and Call Admission Control for inelastic traffic are likely to become necessary in the future.

Two approaches to prevent network congestion will be examined. The first approach is the implementation of a Call Admission Control for inelastic traffic. Since the Internet is too large for a centralized Call Admission Control, an approach based on congestion signals, yielding a distributed Call Admission Control is presented here. Such a Call Admission Control will be further described in Section 7.2.

Sometimes a viewer of a video stream could prefer changes in quality over a blocking probability at the beginning of the stream. Thus, a congestion control may be useful where the data rate of the stream can be changed, for example by changing the quality of the stream. Usually this results in a step utility function with several steps. The application of Congestion Pricing theory to congestion control for inelastic streams will be presented as a second approach in Section 7.3.

7.2 Call Admission Control for Inelastic Traffic

7.2.1 Background

Call Admission Controls are well known from telephony networks and from ATM. For IP-networks, there also exists the *Integrated Services (IntServ)* framework [BCS94] that makes use of the *Resource Reservation Protocol (RSVP)* [ZBHJ97]. Using this framework, senders can ask the routers for resources, which will then be guaranteed for the duration of the connection if the reservation is successful. However, applications and routers must support RSVP, which currently is not the case. Implementation of IntServ requires a turnaround from the original policy to keep network core elements and routers very simple. The relative simplicity of the routers was a major contributor to the fast and successful growth of the Internet. Additionally, IntServ routers would have to manage state information and reservations for several thousands of flows. This immediately raises the question of scalability. The necessary major architectural changes and open scalability issues prevented the broad implementation of the IntServ framework on the Internet. IntServ is mostly employed in local networks.

Thus a different approach to Call Admission Control is proposed. In the previous chapters it was shown that the Congestion Pricing framework is effective in controlling the load on a network. The same framework can be used to decide whether a new flow will be accepted or not. Since the previously presented implementations of Congestion Pricing utilize only a single bit for path price transport, and because there is a delay until the path price is signaled to the source, this approach will be less accurate than a centralized Call Admission Control. On the other hand, this type of Call Admission Control can be implemented in each source, thus establishing a distributed Call Admission Control. This is important for scalability. Furthermore, no bandwidth reservation protocol is necessary. The distributed Call Admission Control could be built into the applications. Alternatively, just like Differentiated Services (DiffServ), a border router could also be the Call Admission Control gateway policing outgoing and incoming flows. While Call Admission Controls based on Congestion Pricing were proposed in the past [KG99, KKZ00], an actual and working implementation will be presented here to demonstrate that the application of Congestion Pricing is feasible.

Two different variants are examined: A passive, *non-probing* Call Admission Control (non-probing CAC), and an active, *probing* Call Admission Control (probing CAC). The passive Call Admission Control monitors the path prices for several paths and stores them. If a new flow is established, the passive, non-probing CAC will read the stored path price for that destination and then decide whether to accept the new flow or not. The probing CAC, on the other hand, will send probe-packets to the destination after detection of a new flow. The destination will echo these probe-packets back, thus providing information on the path price. The disadvantage of the probing CAC is the additional delay introduced by the probing, however, it will provide current information on the congestion state of the path. The non-probing CAC, on the other hand, can utilize already stored information to make an immediate decision. On the other hand, the stored information might be outdated. In practice, both types could be combined such that the network is only probed when the stored information on path prices is older than a certain threshold. Probing CACs have been proposed before, for example in [Kel01, BKS$^+$00]. A discussion of different approaches is given in [AP03]. While most of the approaches include several performance measurements such as loss probability and queuing delay, in this dissertation the focus will be on the marks from the Active Queue Management algorithm indicating the congestion price. It will be shown that this is sufficient for an effective CAC. Further, a probing CAC can be placed at each source, thus no additional signaling or flow detection is necessary. While a non-probing CAC can also be placed at each source, it is most advantageous as gateway CAC where it can monitor several flows and therefore gather and re-use information for several sources (cf. Figure 7.2.1). However, a gateway CAC must be able to detect a new flow. Simple

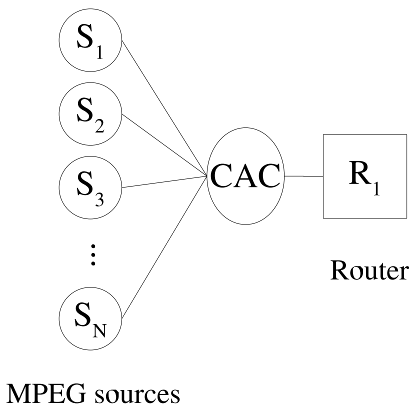


Figure 7.2.1: Call Admission Control gateway

timeout strategies can be used to distinguish different streams from the same source to the same destination in case of video streams because it is valid to assume that a video stream has ended if no frames have been sent for more than a certain amount of time. For example, at a frame rate of 25, a packet should be sent on average every 40 ms. This is different for other types of traffic such as HTTP. While such a CAC gateway could theoretically also be applied to elastic traffic such as HTTP, a web browsing session usually consists of several page requests and variable-length pauses in between. Since a user would not like a page request to a web server to be blocked after he has already received some pages, additional algorithms have to be implemented to detect HTTP requests that belong to the same session. The author of this dissertation proposes to use the *Referer-Header* that contains the last page visited. If the Referer-Header contains the same server name as the current request, the new request will be considered part of the web browsing session that was already accepted. If sessions can be distinguished, the proposed CAC can be used for any type of traffic. Nonetheless, since elastic traffic can already be covered by the previously described congestion control, a CAC should primarily be useful for inelastic traffic types.

7.2.2 Implementation

To demonstrate practical feasibility, both types of Call Admission Controls, non-probing and probing CAC, were implemented using the *Ptolemy Classic* network simulator [PCB]. A bidirectional single bottleneck link model was used to simplify routing (cf. Figure 7.2.2). Only

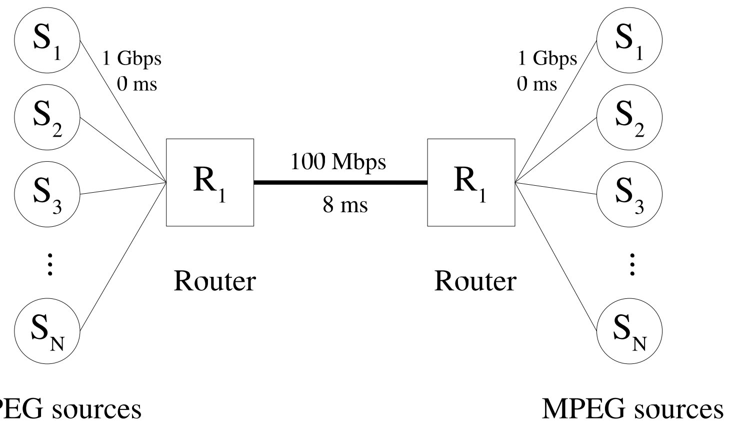


Figure 7.2.2: Single bottleneck link topology

basic results will be presented here to demonstrate the feasibility of such a CAC. Thorough examinations including multi-link topologies were performed in [AP03].

A commonly used standard for digital video compression is ISO MPEG-4 [Gro99]. It allows the choice of different qualities when encoding the source. MPEG-4 video streams were implemented using trace files provided by [FR01]. Every 40 ms a frame from the trace file is read, then a RTP header is added, and the packet is sent via a UDP stack, which fragments the frame if necessary. To model different users, a Poissonian call generator activates such an MPEG source whenever a new call is started. The duration of the MPEG streaming phase is chosen randomly using an exponential distribution. The sources are designed such that they are independent.

Congestion signals generated by a RED queue are used to determine the blocking probability. RED queues were chosen because they are already readily available on the Internet. While RED queues can also be used within a Congestion Pricing framework, a better choice would be an algorithm that was specifically designed for this purpose, such as SBRM. Here it will only be demonstrated that a distributed Call Admission Control is feasible and can be effective and efficient. The optimization of the source and link algorithms is left for further studies.

The non-probing CAC would simply record the average marking probability and block rates if it is above a certain threshold. This can be derived from a logarithmic utility function where $U'(x) = \frac{w}{x} = p$ (cf. Formula 3.2.7) in equilibrium. Since this is not elastic traffic, a logarithmic utility function cannot directly be applied. Therefore, the admission criterion is designed such that a flow is allowed if $p < \frac{w}{x}$, where p is the marking probability, and $\frac{w}{x}$ is the admission threshold. Thus a flow is allowed if an elastic traffic source had to increase its transmission rate to reach that equilibrium, and it is blocked if an elastic source had to decrease its transmission rate. Similarly, the probing CAC would measure the average marking probability by sending out probe packets. In [AP03], a call will be blocked if at least one of the probe packets is marked. Thus, the admission threshold $\frac{w}{x} = \frac{1}{numberOfProbePackets}$. The number of probe packets is therefore inversely proportional to the *willingness to pay*. By choosing the number of probe packets, weights can be given to each source. From Congestion Pricing theory, the inverse blocking rates should then be distributed according to the weighted proportional fairness criterion. This is demonstrated in [AP03].

7.2.3 Simulations and Results

The average data rate of the MPEG streams is 24.36 kBytes/s. Since the bottleneck link's capacity was chosen to be 100 Mbps, about 513 streams can be supported at the same time. A realistic target load of this link should be less than the maximum capacity to avoid persistent or strongly fluctuating queuing delays. In this case, about 90% of the maximum capacity, i.e. a target load of 460 concurrent streams, was chosen. An *effective* CAC should thus not allow more than 460 streams. An *efficient* CAC should on average not block a new call if less than 460 streams are active. The ideal blocking probability for this setup is given by the Erlang-B formula. The proposed CAC is therefore effective if its blocking probability is never above the blocking probability given by the Erlang-B formula. It is efficient, if it matches the Erlang-B formula.

The blocking rates for the passive, non-probing CAC, and for the active, probing CAC are shown in Figure 7.2.3 in comparison to Erlang-B. Both non-probing CAC and probing CAC

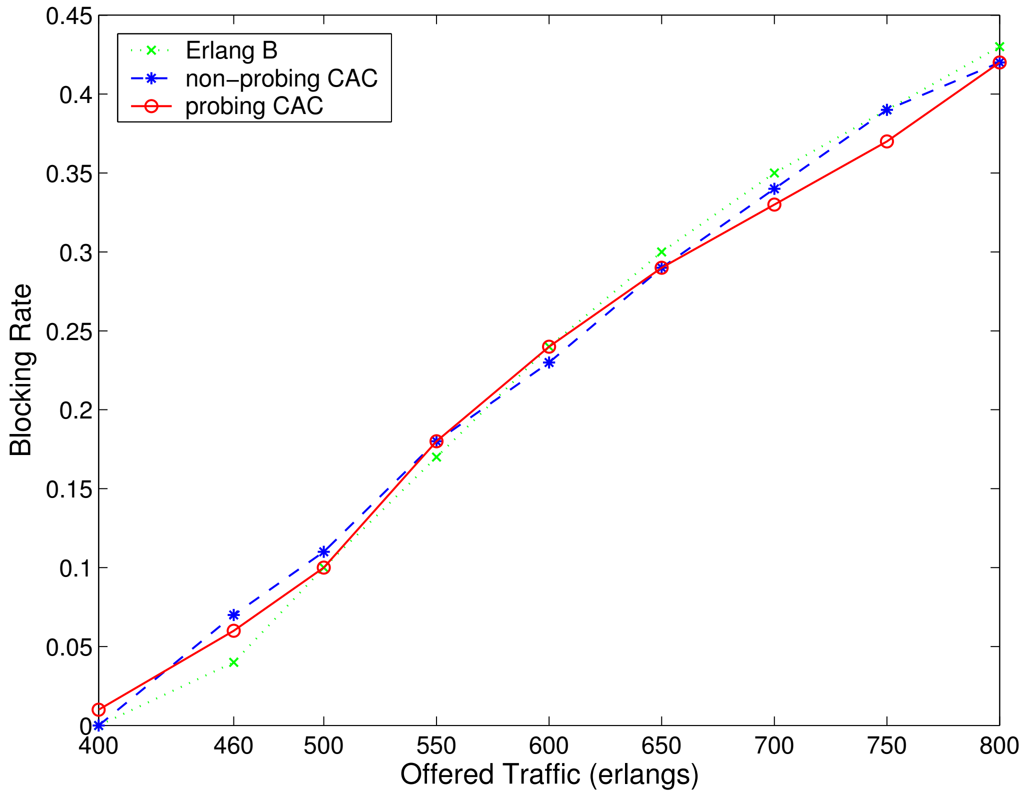
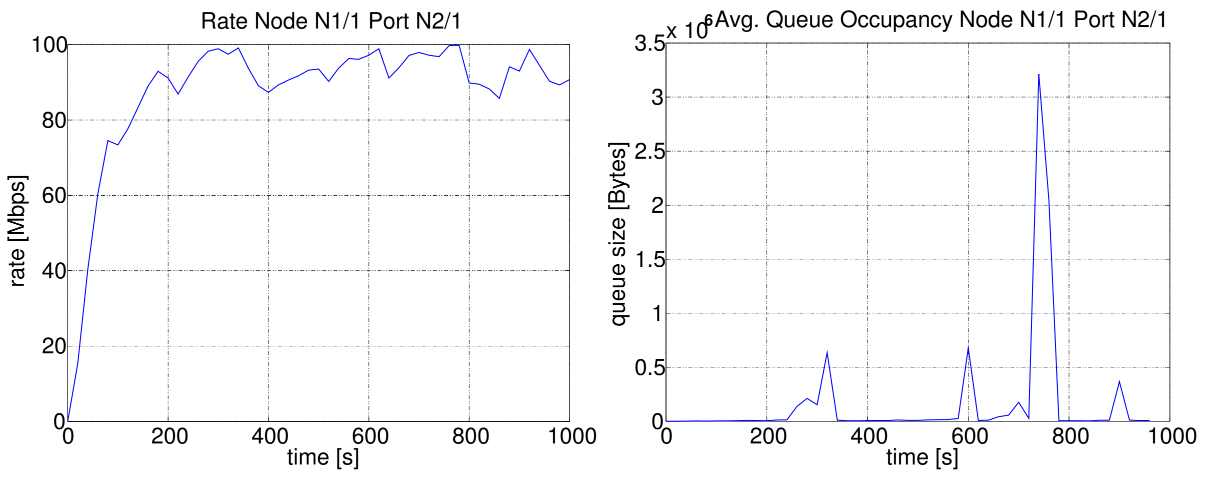


Figure 7.2.3: Comparison of blocking rates with Erlang-B

generate blocking rates close to the theoretical values given by Erlang-B. However, there is a tendency that the blocking rate is too high for loads up to 550 Erlang (over control) and too low for very high loads above 600 Erlang (insufficient control). Since a network should be dimensioned in such a way that very high overload is improbable, the focus will be on the range up to 10% average overload equaling to roughly 560 Erlang. Within this range the blocking rate is slightly higher than the optimal Erlang-B blocking rates. Thus, the proposed distributed CAC algorithms are effective in preventing overload. The distributed CACs can therefore be used to replace a centralized optimal CAC algorithm for the prevention of network overload. The trade-off is a slight reduction in efficiency. In the worst case, average throughput is reduced to 85% [AP03]. Since scalability is extremely important for the Internet, this loss of efficiency is considered tolerable. This is valid even more as without CAC an average throughput of 100% cannot be achieved either.

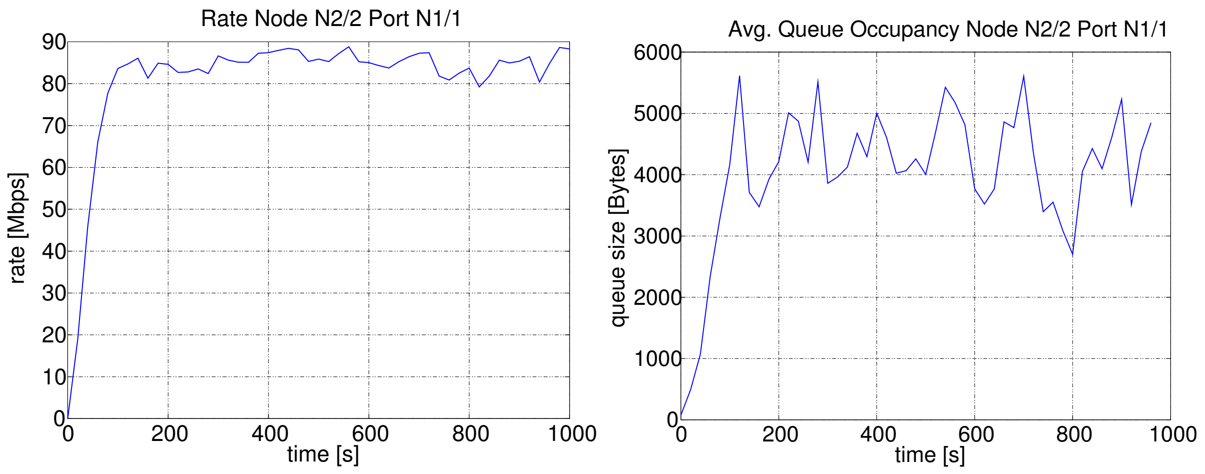
Figure 7.2.4 shows throughput and average queue size over time at a load of 460 Erlang if no Call Admission Control is used. The same scenario, but with non-probing Call Admission



(a) Bottleneck link utilization (b) Queue length

Figure 7.2.4: Without call admission control (load: 460 Erlang) [AP03]

Control is shown in Figure 7.2.5. While the offered load is on average lower than the maximum



(a) Bottleneck link utilization (b) Queue length

Figure 7.2.5: With non-probing call admission control (load: 460 Erlang) [AP03]

supported load of roughly 513 Erlang, the lack of a CAC leads to temporary over-load conditions. These cause a significant increase of the average queue size, temporarily growing above 3,000,000 Bytes (cf. Figure 7.2.4b). At a bottleneck capacity of 100 Mbps, this corresponds to an average additional delay of more than 240 ms which is not tolerable for interactive multimedia streams. Even worse, had there been a limit on the maximum queue length, a large amount of packet loss would have occurred. In both cases, all users of that link would have been negatively impacted. Additionally, strong variations in queuing delay have to be compensated by a large playback buffer. With CAC, on the other hand, the average queue size never grows above 6000 Bytes, corresponding to an acceptable queuing delay of half a millisecond. Utilization, on the other hand, is on average only slightly less than without CAC and varies less (cf. Figures 7.2.4a and 7.2.5a). Thus, a distributed CAC can effectively prevent packet loss and network congestion while keeping utilization nearly constant at a high value.

More complex network topologies, where different clients received video streams from different servers using different paths, were examined in [AP03]. In a parking lot topology with four bottleneck links, the distributed CAC also worked effectively and nearly efficiently. Using a probing CAC, the utilization of all bottleneck links was on average roughly 92% [AP03]. Different admission thresholds were used for different sources. Since Congestion Pricing theory was applied, the blocking rates were distributed such that weighted proportional fairness was achieved. This was explained in Subsection 7.2.2. Thus, by choosing the *willingness to pay*, certain streams can be prioritized analogously to congestion control for elastic traffic.

7.3 Rate-adaptive MPEG Streaming

7.3.1 Motivation

For the encoding of video streams, several video codecs are available today. Most of them compress the original digital information, for example by only transmitting differences between two consecutive picture frames. While this does not lead to loss of information as long as the previous picture frame is known, there are also several video codecs that utilize lossy compression. Depending on the compression level desired, the quality of the video stream will become worse. Thus, video streams are not entirely *inelastic*. If the viewer is willing to accept a lesser quality in case of network congestion, an otherwise necessary Call Admission Control (CAC) could be replaced by some real-time rate-adaptation algorithm that responds to changes in network congestion. This second approach will be examined in this section.

7.3.2 Implementation

Approach

Generally, there are different ways to change compression levels and thus transmission rate and quality of a video stream. These are:

- Adaptive encoding
- Hierarchical encoding

- Switching between pre-encoded video streams

When using *adaptive encoding*, the compression levels are changed on-the-fly for every recipient of the video stream according to the network load [BT94]. This method, however, requires strong computational power and thus does not scale well. While this method might be feasible with specialized hardware or only a single stream, it cannot be used for video servers that serve a large number of clients. This disadvantage is addressed by the *hierarchical encoding*. With this method, the video stream is encoded in several layers, where each layer adds more detailed information and thus improves quality [RHE00]. In case of congestion, only the lower level layers are transmitted. If the network load goes down, more additional layers will be added improving the quality of the stream. However, the additional layers are only useful if the underlying layers are received correctly and completely. Furthermore, this method is still very complex, requires much computational power and can only be used with special codecs.

The third option is the switching between several pre-encoded video streams of the same video, but using different qualities. This requires more storage capacity on the server, but only in relation to the number of videos offered and not in relation to the number of users. Thus, this method will scale well. Today this is a commonly used approach. Before starting the video stream, the user is asked what type of access technology he uses. He can commonly choose between a high speed connection (LAN), a medium speed connection (ADSL, SDSL), or a low speed connection (Modem, ISDN). Depending on the answer, a different quality will be presented. In this example, however, this decision by the receiver is made only once — before the stream is started — and under the assumption that the access link is the limiting bottleneck.

Each of the three methods described before can be used for congestion control for video streams. The third approach is most promising, since switching between pre-encoded video streams seems the most feasible and acceptable method in current networks. Nonetheless, the results obtained and problems described later in this section also apply to the other methods.

Video Sources

The same MPEG-4 sources that were presented in Subsection 7.2.2 are used here. Whenever the next frame to be sent is a full-size frame containing complete information of the next picture ("P frame"), a choice is made whether the transmission rate should be changed and a different trace file with a different quality of the same video stream should be used.

Quality Selection

The decision regarding which quality is to be used is based on the average transmission rates of each quality and the available bandwidth on the network according to the Congestion Pricing algorithm. Since three different quality levels are used here with one corresponding transmission rate each, the utility function applicable in this case is shown in Figure 7.3.1. Obviously, such a utility function violates the concavity requirement (cf. Subsection 3.2.1). For this reason, the target rate was determined analogously to SBRM (cf. Section 5.4) using a logarithmic utility function:

$$x_{target}(t+T) := x_{target}(t) + \kappa T \left(w - p(t)x_{target}(t)\right), \qquad (7.3.1)$$

where T is the time between two updates and κ is the gain per update interval, p is the path price, w is the willingness to pay. Note that this rate update rule is identical to (4.2.1).

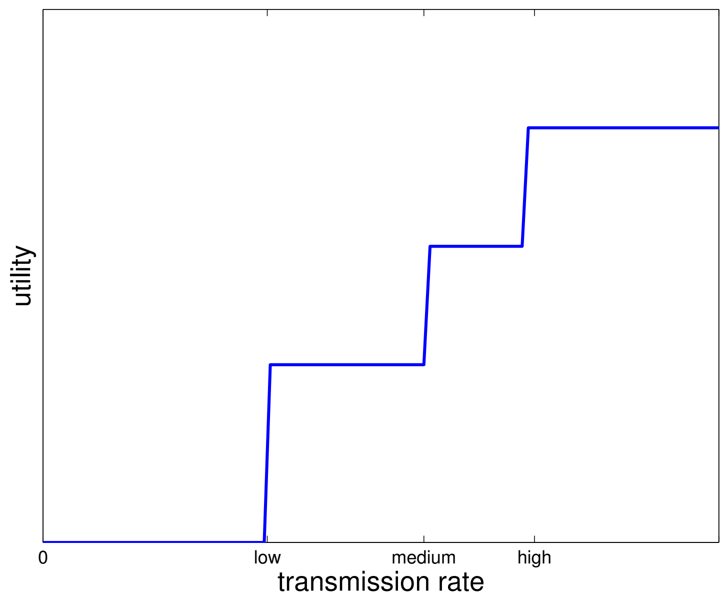


Figure 7.3.1: Step utility function

The quality is then switched as follows:

$$\begin{array}{rll} \text{if} & x_{target} > x_{betterQuality} \cdot f_{incr} & \text{then select better quality} \\ \text{else if} & x_{target} < x_{currentQuality} \cdot f_{decr} & \text{then select worse quality} \\ \text{else} & & \text{keep current quality} \end{array} \tag{7.3.2}$$

where f_{incr} is a factor describing by how much the target rate must exceed the rate of the next better quality before increasing quality, and f_{decr} is a factor describing by how much the target rate must be lower than the rate of the current quality before decreasing quality. A variant was also implemented that completely stops transmissions when the target rate becomes lower than the worst available quality.

Marking and Price Feedback

Since the calculation of the target rate depends on the pricing information given by the network, a mechanism must be implemented to signal that information to the source. UDP lacks a retransmission algorithm and therefore does not provide reliable transfer. For this reason, packet loss and loss of pricing feedback information must also be taken into account. As a solution, counters were implemented at the source and sink. The source counts total packets sent, the sink counts total packets received and number of packets marked. These two counters are then sent back to the source in regular RTP quality reports. From these counters, the source can calculate the fraction of packets marked and the fraction of packets lost. For the calculation of the target rate no difference is made on whether a packet was marked or lost. It would be easy, however, to implement a different reaction to packet loss than to packet marks. The path price p is thus calculated as follows:

$$p = \frac{(num_{sentPackets} - num_{receivedPackets}) + num_{markedPackets}}{num_{sentPackets}}. \tag{7.3.3}$$

7.3.3 Simulations and Results

Setup

Simulations were performed using the *Ptolemy Classic* [PCB] simulator. A single bottleneck link topology shown in Figure 7.3.2 was used to evaluate performance of the rate control. Three

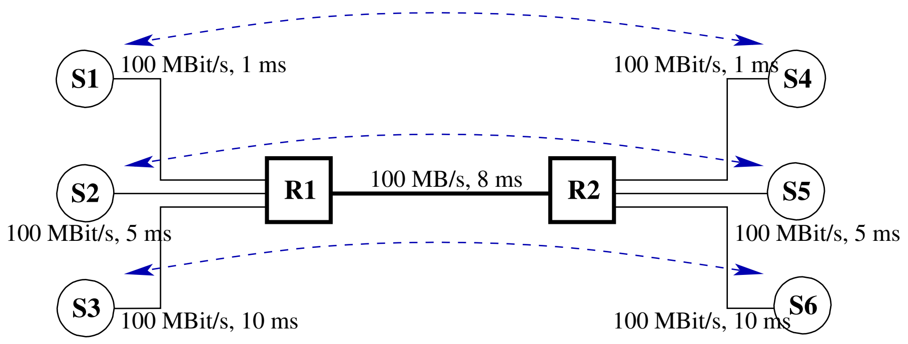


Figure 7.3.2: Single bottleneck link topology with three different round-trip delays

paths share the same bottleneck link, but use different delays, resulting in three different round-trip delays: 20 ms, 36 ms, and 56 ms. At each source node (S1, S2, S3) twenty video streams are constantly served. The parameters of the algorithms are shown in Table 7.3.1. Again, RED

Table 7.3.1: Simulation Parameters

Source algorithm			Queue algorithm (RED)	
Parameter	Value		Parameter	Value
WTP w_n	$20000\frac{bytes}{s}$		th_{min}	6000 bytes
κ	0.005		th_{max}	60000 bytes
$rate_{weight}$	0.1		p_{max}	0.4
f_{incr}	1.1		q_{weight}	0.01
f_{decr}	0.95		gentle	true

gateways are used for the simulations because they are more common on the Internet today. For optimal results, SBRM gateways could be used instead.

Overall Load (Effectiveness)

To measure the effectiveness of the rate control, the offered load was changed by increasing the number of MPEG-4 sources. The average queue size was then measured as an indicator for the actual load of the network. Figure 7.3.3 shows the average queue size as a function of the number of active sources. Three different curves are shown. The first plot applies if no rate control is used. As expected, the average queue size as an indicator for the actual network load grows nearly linearly to the offered load. The second plot shows MPEG sources using rate control, but without interruptions of the stream. For this reason, each source will transmit at least at the minimum quality. This mechanism can reduce the actual network load, but it will still grow as each additional source will add more packets injected into the network. Finally,

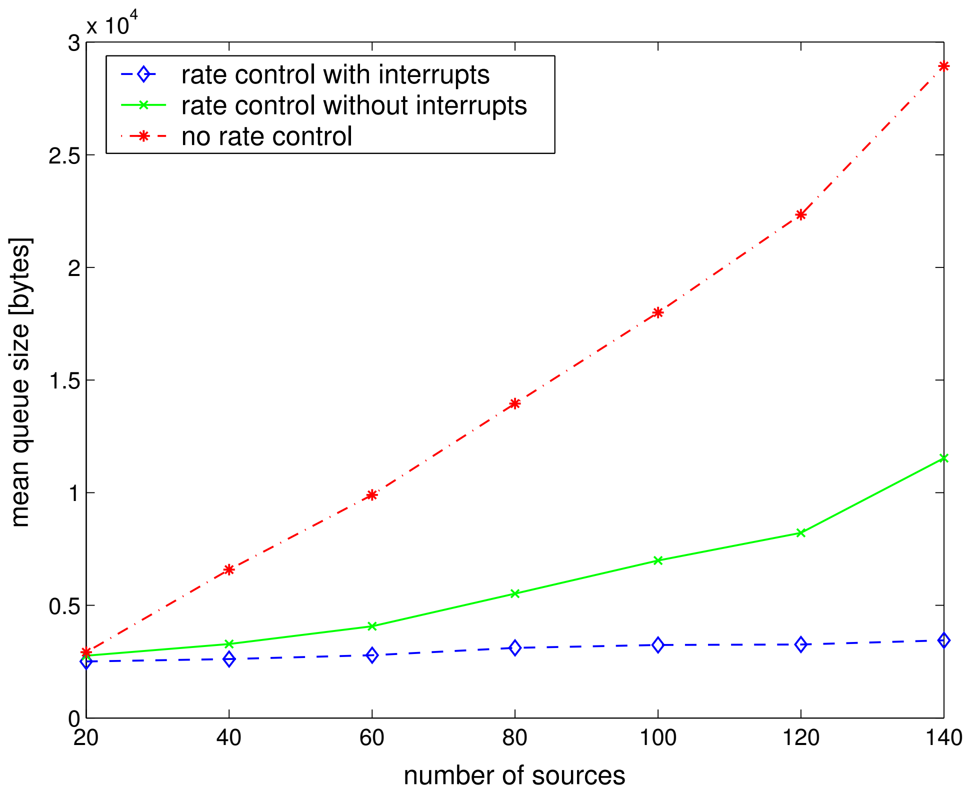


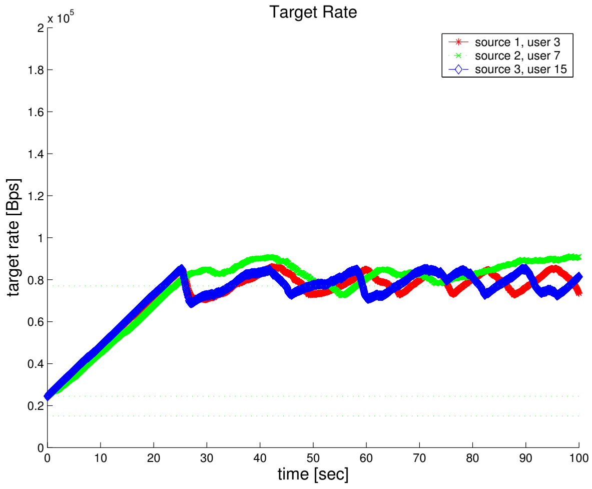
Figure 7.3.3: Network load (mean queue size) as function of offered load (number of sources)

the third plot displays sources that are interrupted if the network load becomes too high. In this case, overloading of the network can effectively be avoided.

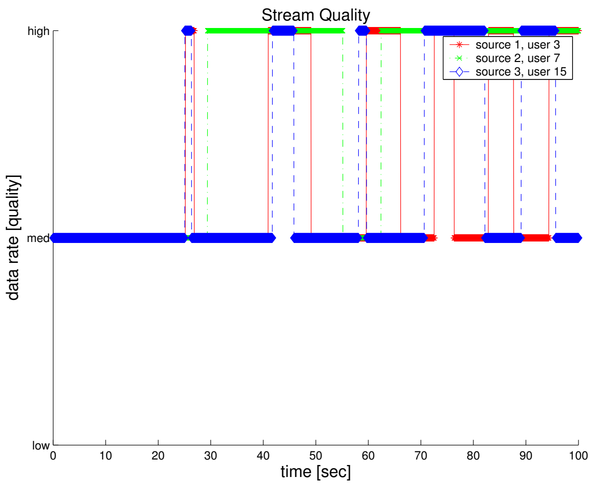
Thus, the proposed rate control algorithm for MPEG streams without interruptions cannot completely avoid network congestion. This would have been possible if stream interruptions were acceptable. Since users do not want unexpected interruptions in their viewing of the video stream, the proposed rate control can only reduce the impact of offered load on the network congestion (second curve in Figure 7.3.3). It is still effective to increase the number of streams supported by the network if demand is high. Combining it with the Call Admission Control (CAC) presented in Section 7.2 is recommended in order to avoid a congestion collapse. This combination could be used by network operators to offer a low-class service to those users willing to accept worse quality if the demand is high in exchange for a higher call admission rate.

Dynamics

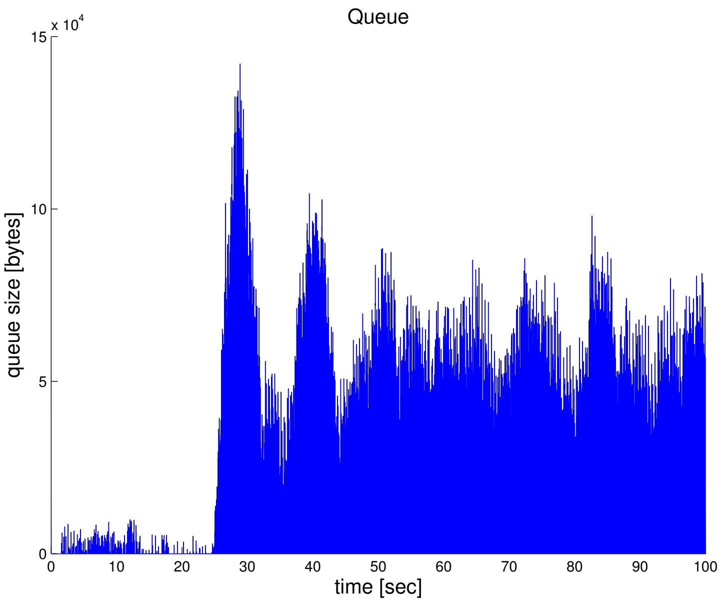
To further evaluate the performance of the rate-control algorithm, the target rate x_{target} as a function of time was recorded for 100 seconds. Additionally, the quality used at any point in time and the average queue size over time were recorded. The results are shown in Figure 7.3.4. From 7.3.4a the algorithm seems to work perfectly. Even though the round-trip time is different for the different sources, the target rate is determined equally. Thus, the algorithm is fair, every source gets roughly the same rate. Also, after an initial phase to allow convergence, the average queue size is almost steady over time (cf. Figure 7.3.4c). However, in order to achieve these



(a) Target rate



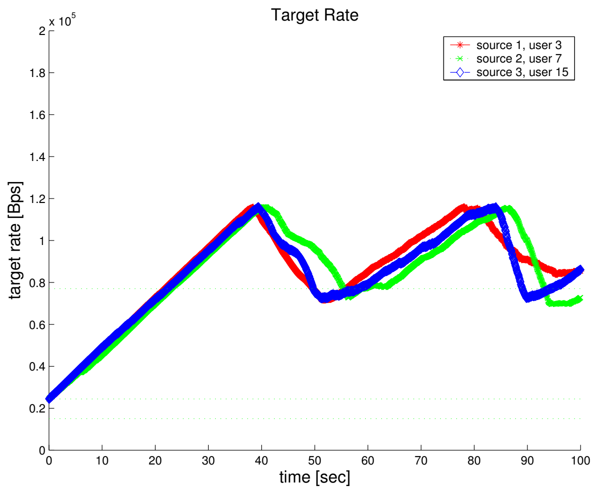
(b) Selected quality



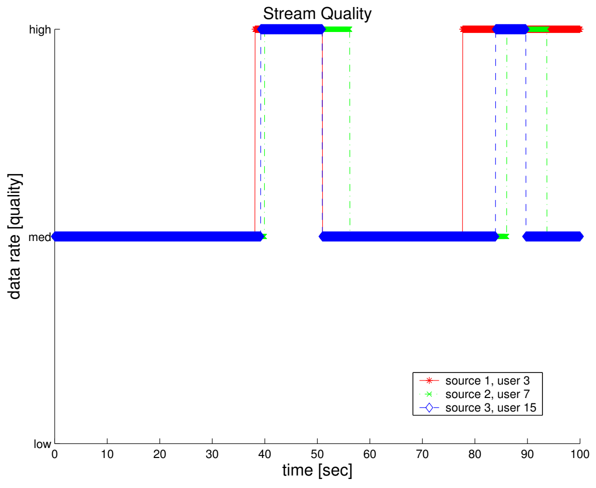
(c) Queue size

Figure 7.3.4: Rate control for MPEG streams with three different qualities

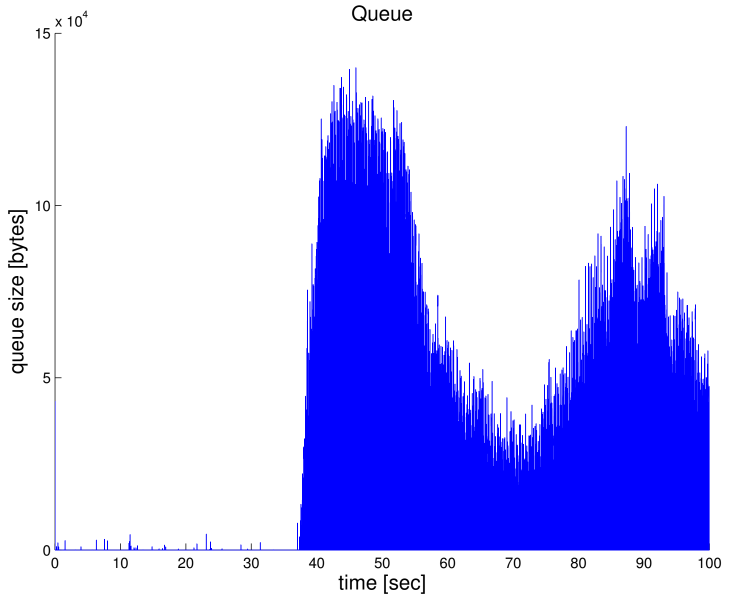
desirable properties, the average target rate has to be maintained by switching between two qualities. This will lead to fluctuating qualities of the video stream (cf. Figure 7.3.4b). It is reasonable to assume that an average viewer would not enjoy such behavior. Flipping between two qualities can be reduced by increasing the factor f_{incr} (cf. Formula 7.3.2). This is shown in Figure 7.3.5. With the modified parameter settings, the target rate has to grow higher before



(a) Target rate



(b) Selected quality



(c) Queue size

Figure 7.3.5: Rate control for MPEG streams with three different qualities and $f_{incr} = 2.0$

the quality is increased (cf. Figure 7.3.5a). This will lead to a reduced number of changes (cf. Figure 7.3.5b). Since response to changing network conditions is reduced, the average queue size will not be constant over time (cf. Figure 7.3.5c). Thus, while this second approach is better for the viewer of a movie as the number of visible quality changes is reduced, from a network operator's perspective, there are notable disadvantages. Therefore a third approach is proposed that will solve both problems at the cost of fairness.

Improved Quality Selection Algorithm (With Random Increase)

The quality selection algorithm (7.3.2) is changed as follows:

$$\begin{array}{llll}
\text{if} & x_{target} > x_{betterQuality} \cdot f_{incr} \text{ and } flag & \text{then select better quality at a probability of 0.5,} & \\
 & & \text{and set } flag = false. & (7.3.4) \\
\text{else if} & x_{target} < x_{currentQuality} \cdot f_{decr} & \text{then select worse quality, set } flag = true & (7.3.5) \\
\text{else} & & \text{keep current quality} &
\end{array}$$

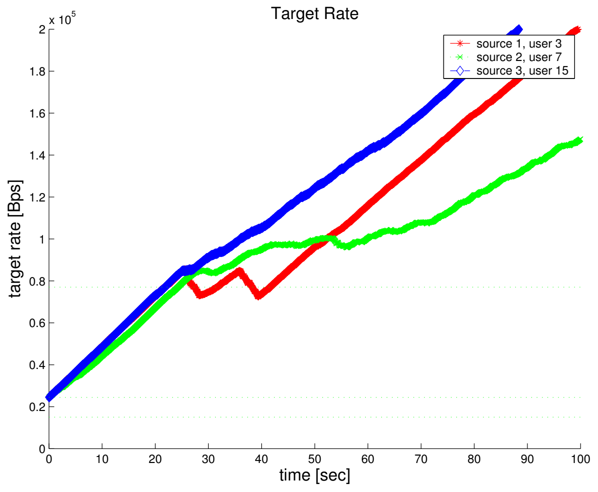
Note that the rate is only increased at a probability of 50% when the target rate x_{target} becomes higher than the rate of the better quality stream $x_{betterQuality}$ times the increase factor f_{incr}. Obviously, this will cause unfairness.

The simulation results are shown in Figure 7.3.6. As visible in Figure 7.3.6b, the number of quality changes is greatly reduced. Furthermore, the mean queue size (cf. Figure 7.3.6c) does not fluctuate as much as before. Thus, this MPEG rate control algorithm (7.3.5) is much better from a user's perspective than the previously suggested algorithm (7.3.2). The disadvantage is unfairness between users. One user (source 2, user 7) receives a better rate than the other users (source 1, user 3, and source 3, user 13).

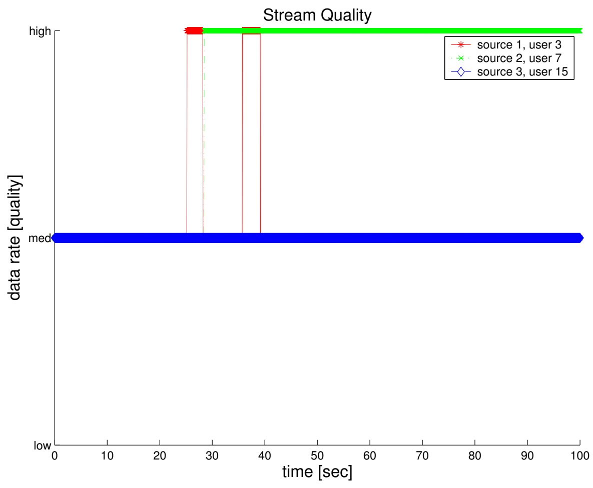
7.4 Conclusions

In this chapter two approaches to avoid congestion in the case of inelastic traffic were presented. The first approach is the implementation of a distributed Call Admission Control. Although further research is needed, the results presented here already prove that a distributed Call Admission Control based on Congestion Pricing theory is feasible. No modifications to the network core devices are necessary, thus deployment on the Internet is possible. It can be implemented at gateway routers or even in the sources themselves to avoid scalability problems. Even though such a distributed CAC does not work as perfectly as a centralized CAC could, it is effective within the interesting load range and very efficient. When used, it keeps utilization and average queue size nearly stable and thus prevents congestion and packet loss. Thus, a distributed Call Admission Control will improve overall performance of the network.

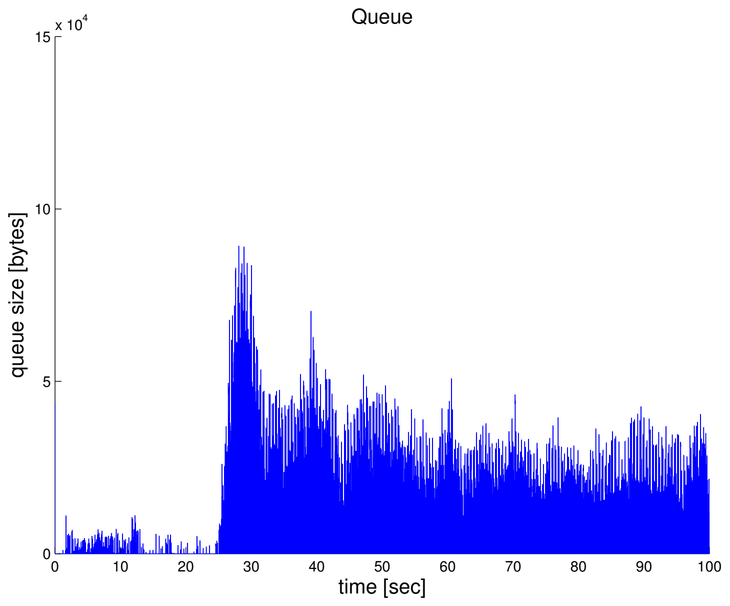
Congestion Pricing can also be applied to rate-control MPEG streams, which is the second proposal. Since streams are usually not elastic, Congestion Pricing will lead to oscillations between two rate steps in order to achieve the desired average rate. While this is effective in avoiding congestion and optimal utilization of network resources, a user would not like the changes in quality that are related to the rate changes. To overcome this problem, a random component was included before the transmission rate could be increased. This will lead to an unfair rate allocation, but avoid the aforementioned oscillations between two rate steps. A Call Admission Control is still necessary, unless rate reductions to zero during the streaming are tolerable.



(a) Target rate



(b) Selected quality



(c) Queue size

Figure 7.3.6: Rate control for MPEG streams with random increase

Chapter 8

Conclusions

In this dissertation, the existing congestion control algorithms for TCP/IP and their problems were presented. Congestion Pricing theory was applied to TCP in an attempt to solve these problems. Congestion Pricing is a mathematical framework derived from economics and game theory, and can be used to optimize traffic loads on a network and thus avoid congestion. The strength of Congestion Pricing is the distribution of the optimization problem to the sources. Furthermore, each user can define his preference for bandwidth. This feature can then be used to implement different service classes without the need for the storage of state information in the network nodes or special scheduling algorithms.

While in theory these are very useful features, the Congestion Pricing framework assumes no delays and also assumes continuous fluid-flow or time-slotted networks. Since this does not apply to packet based networks such as the Internet, a specific adaptation had to be developed to be able to apply Congestion Pricing to TCP. Using this adaptation, which was named "CP-TCP/EPF", it was shown that the congestion control algorithm can be improved to better utilize the available resources, maintain low queue sizes, and prioritize streams according to the settings. "CP-TCP/EPF" by far outperforms current TCP variants, and can better adapt to changing network conditions. Thus, oscillations, inherent to conventional TCP with drop-tail queues can be avoided. CP-TCP/EPF is therefore a base for scalable and efficient congestion control.

However, even though CP-TCP/EPF works in packet based networks, it requires changes to the IP header and every router. Since IP is the fundamental protocol of the Internet, it is very improbable that it can be modified. Similarly, replacement of all routers cannot be expected. Instead, a way was presented to deploy Congestion Pricing based TCP on the Internet with minimal modifications. A main problem is the transportation of the feedback signals, since current standards only allow a single bit for congestion information in the IP header. Different proposals to encode congestion prices in a single bit were presented and compared by means of simulation. The insights from these simulations lead to the development of a new algorithm, the Single Bit Resource Marking (SBRM) proposal. SBRM only needs a single bit for the encoding of the congestion prices, but still provides all significant characteristics of Congestion Pricing theory. Simulations have shown, that SBRM — just like CP-TCP/EPF — can establish high utilization of the bottleneck link with low queue sizes. It reacts well to changing network conditions, and its parameters work well over a wide range of network conditions. And, of course, it by far outperforms any conventional TCP variant. SBRM therefore also provides scalable and efficient congestion control. Making some minor additional changes to SBRM, it can be employed with current TCP receivers. Only the sending side — usually the servers — must be modified. This fully compatible variant was named "TCP/RM", and it was demonstrated that it even works with drop-tail and RED queues that are commonplace on the Internet today. TCP/RM is the fastest and easiest way to bring the advantages of Congestion Pricing to the Internet.

The application of control theoretic models to congestion control algorithms is a relatively new field of research. Such a model was also developed for SBRM to examine linear stability. SBRM is stable in most network conditions. Its stable areas of operation are much larger than those of conventional TCP. However, SBRM can also become unstable under certain conditions. Using the control theoretic model, additional modifications to SBRM can be made in the future to ensure stability for all possible modes of operation. Nonetheless, even when SBRM is unstable, it yields good results. Though a drop in link utilization can be observed, persistent congestion is still avoided. In this dissertation a worst case scenario with constant demands and synchronized sources was examined, in an actual network the impact of instability will be lesser still. In the opinion of the author of this dissertation a greater problem still to solve is the start-up phase of new flows, which is important since most flows are still short-lived (e.g. HTTP traffic).

Congestion Pricing can also be applied to multimedia streams. Although theoretically the Congestion Pricing framework does not apply because a concave user's utility function cannot be found for inelastic traffic, this problem was solved by working with a virtual "target rate". This can then be used as the admission threshold for a distributed Call Admission Control or to select different qualities of the multimedia stream. Since the share of inelastic traffic on the Internet is slowly increasing, these approaches are promising for the future. However, problems with the application of Congestion Pricing theory associated with inelastic traffic were also shown. These problems can be solved by abandoning the requirement of fair and equal rate distribution between similar users. Future research is still necessary to improve these algorithms.

The approaches and algorithms presented in this dissertation are focused on congestion control for the Internet, but can be applied to any kind of network or where the use of resources has to be optimized. The use of Congestion Pricing information for routing decisions is also anticipated. Many routing algorithms consider "link costs" when they make a routing decision. Basing these link costs on Congestion Pricing theory could lead to better utilization and also to the possibility of using different classes of service depending on the user's demands.

Also, peer-to-peer applications are growing in demand. The operation of some servers is at present very costly due to the amount of traffic (for example Usenet servers), in such as case peer-to-peer networks might be used as replacement for classical client-server based applications in the future and therefore gain even higher shares on the Internet. While the presented algorithms can already be used to control the transmission rates between peers when transmitting files, Congestion Pricing can further be used to "route" traffic or requests for file fragments between peers. Further, at the top layer it could be used to decide which clients are prioritized.

The future will also introduce completely new applications to communication networks. Optimization will be important for almost all of them, and Congestion Pricing theory is applicable. For example, currently Voice-over-IP is growing strongly, and the introduction of Call Admission Controls is the natural next step to avoid quality problems. Voice traffic should be prioritized over the data because it is delay critical. As was shown in this dissertation, Congestion Pricing can be used to solve both problems in a decentralized and thus scalable and simple way.

Bibliography

[AL00] S. Athuraliya and S.H. Low. Optimization flow control, II: Implementation. Technical report, Melbourne University, May 2000. Available at: http://netlab.caltech.edu/pub/papers/rem2.ps.gz.

[All03] M. Allman. Tcp congestion control with appropriate byte counting (abc). RFC 3465, Internet Engineering Task Force, February 2003.

[ALLY01] S. Athuraliya, V.H. Li, S.H. Low, and Q. Yin. REM: Active queue management. In J.M. de Souza, N.L.S. da Fonseca, and E.A. de Souza e Silva, editors, *Teletraffic Engineering in the Internet Era, Proc. of the International Teletraffic Congress – ITC-17, Salvador da Bahia, Brazil*, volume 4 of *Teletraffic Science and Engineering*, pages 817–828. Elsevier, September 2001.

[AP03] Q. An Pham. Distributed call admission control for packet-switched networks. Project work, Technical University Hamburg-Harburg, Department of Communication Networks, June 2003.

[APS99] M. Allman, V. Paxson, and W. Richard Stevens. TCP congestion control. RFC 2581, Internet Engineering Task Force, April 1999.

[BBC$^+$98] S. Blake, D. L. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated service. RFC 2475, Internet Engineering Task Force, December 1998.

[BCC$^+$98] B. Braden, D. D. Clark, Jon Crowcroft, B. S. Davie, S. E. Deering, D. Estrin, S. Floyd, and V. Jacobson. Recommendations on queue management and congestion avoidance in the Internet. RFC 2309, Internet Engineering Task Force, April 1998.

[BCS94] R. Braden, D. D. Clark, and S. Shenker. Integrated services in the Internet architecture: an overview. RFC 1633, Internet Engineering Task Force, June 1994.

[Bel03] Kai Below. *Methodology for Parameterisation of Large Scale Network Simulations*. PhD thesis, Technical University Hamburg-Harburg, Hamburg, Germany, December 2003.

[Bid] M. Biddiscombe. Proportional fairness – a query. Unpublished. Avaliable at: http://www.users.globalnet.co.uk/~priatel/martin/work/propfair.ps.

[BK02] K. Below and U. Killat. On the configuration of simulations of large network models with HTTP/TCP sources. In P. Tran-Gia and J. Roberts, editors, *15th ITC Specialist Seminar, Internet Traffic Engineering and Traffic Management*, pages 143–149, Würzburg, July 2002.

[BKS$^+$00] Lee Breslau, Edward Knightly, Scott Shenker, Ion Sotoica, and Hui Zhang. Endpoint admission control: Architectural issues and performance. In *SIGCOMM2000*, pages 57–69, 2000.

[BOP94] Lawrence S. Brakmo, Sean W. O'Malley, and Larry L. Peterson. TCP vegas: New techniques for congestion detection and avoidance. In *SIGCOMM*, pages 24–35, 1994.

[BP95] L. Brakmo and L. Peterson. TCP vegas: End to end congestion avoidance on a global internet. *IEEE Journal on Selected Areas in Communication*, 13(8):1465–1480, October 1995.

[Bra89] R. Braden. Requirements for Internet hosts - communication layers. RFC 1122, Internet Engineering Task Force, October 1989.

[BSMM99] I.N. Bronstein, K.A. Semendjajew, G. Musiol, and H. Mühlig. *Taschenbuch der Mathematik*. Verlag Harry Deutsch, 4th edition, 1999.

[BT94] Jean-Chrysostome Bolot and Thierry Turletti. A rate control mechanism for packet video in the internet. In *INFOCOM (3)*, pages 1216–1223, 1994.

[Büc01] M. Büchling. Entwurf und Analyse einer TCP-ECN-Implementierung für den Ptolemy-Simulator. Project work, Technical University Hamburg-Harburg, Department of Communication Networks, July 2001.

[Cla82] D. D. Clark. Window and acknowledgement strategy in TCP. RFC 813, Internet Engineering Task Force, July 1982.

[FAS03] Fast AQM Scalable TCP (FAST). NETLAB: California Institute of Technology. Web Site, July 2003. http://netlab.caltech.edu/FAST/.

[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.

[FGM$^+$99] R. Fielding, J. Gettys, J. C. Mogul, H. Frystyk, L. Masinter, P. J. Leach, and T. Berners-Lee. Hypertext transfer protocol – HTTP/1.1. RFC 2616, Internet Engineering Task Force, June 1999.

[FGS01] S. Floyd, R. Gummadi, and S. Shenker. Adaptive RED: An algorithm for increasing the robustness of RED's active queue management. Available at: http://www.icir.org/floyd/adaptivered/, August 2001.

[FH99] S. Floyd and T. Henderson. The newreno modification to tcp's fast recovery algorithm. RFC 2582, Internet Engineering Task Force, April 1999.

[FJ93] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, August 1993.

[FKSS99] W. Feng, D. Kandlur, D. Saha, and Kang G. Shin. BLUE: A new class of active queue management algorithms. Technical Report CSE-TR-387-99, University of Michigan, 15, 1999.

[Flo03] S. Floyd. Highspeed TCP for large congestion windows. RFC 3649, Internet Engineering Task Force, December 2003.

[FPL03] J. Doyle F. Paganini and S. Low. *Multidisciplinary Research in Control: The Mohammed Dahleh Symposium 2002.*, chapter A Control Theoretical Look at Internet Congestion Control. Number 289 in Lecture Notes in Control and Information Sciences. L. Giarre' and B. Bamieh, Springer Verlag, Berlin, 2003.

[FR01] F. Fitzek and M. Reisslein. MPEG–4 and H.263 Video Traces for Network Performance Evaluation. *IEEE Network*, 15(6):40–54, November/December 2001. Video traces available at http://www-tkn.ee.tu-berlin.de/research/trace/trace.html.

[GK99] R.J. Gibbens and F.P. Kelly. Resource pricing and the evolution of congestion control. *Automatica 35*, pages 1969–1985, 1999.

[Gro99] Moving Pictures Expert Group. Mpeg-4 standard (iso/iec 14496), 1999.

[Ham01] T. Hamann. Dynamics of different congestion pricing strategies. Master's thesis, Technical University Hamburg-Harburg, Department of Communication Networks, May 2001.

[HMTG01a] C. V. Hollot, V. Misra, D. F. Towsley, and W. Gong. A control theoretic analysis of RED. In *INFOCOM*, pages 1510–1519, 2001.

[HMTG01b] C. V. Hollot, V. Misra, D. F. Towsley, and W. Gong. On designing improved controllers for AQM routers supporting TCP flows. In *INFOCOM*, pages 1726–1734, 2001.

[HW00] T. Hamann and J. Walrand. A new fair window algorithm for ecn capaple tcp (new-ecn). In *Proceedings of the Infocom 2000*, volume 3, pages 1528–1536, March 2000.

[IB01] O. Imer and T. Basar. Control of congestion in high-speed networks. *European Journal of Control*, 7:132–144, September 2001.

[ISC03] Internet Software Consortium. Internet domain survey, January 2003.

[JB88] V. Jacobson and R. Braden. TCP extensions for long-delay paths. RFC 1072, Internet Engineering Task Force, October 1988.

[JBB92] V. Jacobson, R. Braden, and D. Borman. TCP extensions for high performance. RFC 1323, Internet Engineering Task Force, May 1992.

[JF02] A. Jain and S. Floyd. Quick-Start for TCP and IP, October 2002. Internet Draft (expired): http://www.icir.org/floyd/papers/draft-amit-quick-start-02.txt.

[JT01] R. Johari and D. Tan. End-to-end congestion control for the internet: Delays and stability. *To appear in IEEE Transactions on Networking.*, 2001.

[Kel97] F.P. Kelly. Charging and rate control for elastic traffic. *European Transactions on Telecommunications*, 8:33–37, January 1997.

[Kel01] T. Kelly. An ECN probe-based connection acceptance control. *Computer Communication Review*, 31(3), July 2001.

[Key01] P. Key. Resource pricing for differentiated services. In U. Killat and W. Lamersdorf, editors, *Proceedings of: Kommunikation in Verteilten Systemen (KiVS) 2001.*, Informatik aktuell. Springer-Verlag, February 2001.

[KF02] M. Kwon and S. Fahmy. Tcp increase/decrease behavior with explicit congestion notification (ecn). In *Proc. of the ICC 2002, New York*. IEEE, April 2002.

[KG99] F.P. Kelly and R.J. Gibbens. Distributed connection acceptance control for a connectionless network. In *Proceedings ITC 16*. University of Cambridge, June 1999. Presented at 16th International Teletraffic Congress.

[KHR00] D. Katabi, M. Handley, and C. Rohrs. Congestion control for high bandwidth-delay product networks. In *SIGCOMM*, August 2000.

[KKZ00] F.P. Kelly, P. Key, and S. Zachary. Distributed admission control. *IEEE Journal of Selected Areas in Communications*, 18(12), December 2000.

[KL86] B. Kantor and P. Lapsley. Network news transfer protocol. RFC 977, Internet Engineering Task Force, February 1986.

[KMT98] F.P. Kelly, A.K. Maulloo, and D.K.H. Tan. Rate control for communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, 49:237–252, 1998.

[KP87] P. Karn and C. Partridge. Improving round-trip time estimates in reliable transport protocols. *ACM/SIGCOMM*, pages 2–7, August 1987.

[LA00] R. La and V. Anantharam. Charge-sensitive TCP and rate control in the internet. *Proceedings of the IEEE Infocom '00*, March 2000.

[LL99] S.H. Low and D. Lapsley. Optimization flow control, I: basic algorithm and convergence. *IEEE/ACM Transactions on Networking*, 7(6):861–874, December 1999.

[Low00] S.H. Low. A duality model of TCP and queue management algorithms. In *Proceedings of ITC Specialist Seminar on IP Traffic Measurement, Modeling and Management, Monterey, CA*, September 2000.

[LPW01] S.H. Low, L. Peterson, and L. Wang. Understanding vegas: A duality model. In *Proceedings of the ACM Sigmetrics*, Boston, MA, June 2001.

[LPW$^+$02] S.H. Low, F. Paganini, J. Wang, S. Adlakha, and J.C. Doyle. Dynamics of TCP/RED and a scalable control. In *Proceedings of the IEEE Infocom 2002, New York*, pages 239–248, 2002.

[LR98] T. Li and Y. Rekhter. A provider architecture for differentiated services and traffic engineering (PASTE). RFC 2430, Internet Engineering Task Force, October 1998.

[LWA99] Richard J. La, Jean Walrand, and Venkat Anantharam. Issues in TCP Vegas. UCB/ERL Memorandum, No. M99/3, Electronics Research Laboratory, University of California, Berkeley, available at: www.eecs.berkeley.edu/~ananth/1999-2001/Richard/IssuesInTCPVegas.pdf, 1999.

[LWG03] C.-N. Long, J. Wu, and X.-P. Guan. Local stability of REM algorithm with time-varying delays. *IEEE Communication Letters*, 7(3), March 2003.

[Mar87] S. L. Marple. *Digital Spectral Analysis*. Prentice-Hall, Englewood Cliffs, NJ, 1987.

[Mas00] L. Massoulié. Stability of distributed congestion control with heterogeneous feedback delays. Technical Report Technical Report 2000-111, Microsoft Research, Microsoft Coporation, 2000.

[MBDL99] M. May, J. Bolot, C. Diot, and B. Lyles. Reasons not to deploy RED. In *Proc. of 7th. International Workshop on Quality of Service (IWQoS'99), London*, pages 260–262, June 1999.

[Mil92] D. L. Mills. Network time protocol (version 3) specification, implementation. RFC 1305, Internet Engineering Task Force, March 1992.

[MLAW99] Jeonghoon Mo, Richard J. La, Venkat Anantharam, and Jean C. Walrand. Analysis and comparison of TCP Reno and Vegas. In *INFOCOM (3)*, pages 1556–1563, 1999.

[MMFR96] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. TCP selective acknowledgement options. RFC 2018, Internet Engineering Task Force, October 1996.

[MMV95] J.K. MacKie-Mason and H.R. Varian. Pricing congestible network resources. *IEEE Journal on Selected Areas in Communications*, 13(7):1141–1149, September 1995.

[Moc87] P. V. Mockapetris. Domain names - implementation and specification. RFC 1035, Internet Engineering Task Force, November 1987.

[OLW99] Teunis J. Ott, T. V. Lakshman, and Larry H. Wong. SRED: Stabilized RED. In *Proceedings of INFOCOM*, volume 3, pages 1346–1355, 1999.

[Pat03] S. Patchotepong. Validation and extension of a control theoretic model for TCP/RED and Single-Bit Resource Marking strategy (SBRM). Project work, Technical University Hamburg-Harburg, Department of Communication Networks, May 2003.

[PCB] Ptolemy classic. University of California at Berkeley. http://ptolemy.eecs.berkeley.edu/ptolemyclassic/body.htm.

[Pos80] J. B. Postel. User datagram protocol. RFC 768, Internet Engineering Task Force, August 1980.

[Pos81a] J. B. Postel. Internet protocol. RFC 791, Internet Engineering Task Force, September 1981.

[Pos81b] J. B. Postel. NCP/TCP transition plan. RFC 801, Internet Engineering Task Force, November 1981.

[Pos81c] J. B. Postel. Transmission control protocol. RFC 793, Internet Engineering Task Force, September 1981.

[Pos82] J. B. Postel. Simple mail transfer protocol. RFC 821, Internet Engineering Task Force, August 1982.

[Pos83] J. B. Postel. TCP maximum segment size and related topics. RFC 879, Internet Engineering Task Force, November 1983.

[PR85] J. B. Postel and J. F. Reynolds. File transfer protocol. RFC 959, Internet Engineering Task Force, October 1985.

[Red02] S. Redenbach. Fairness- und TCP-Friendliness-Verhalten verschiedener TCP-REM-Implementierungen. Project work, Technical University Hamburg-Harburg, Department of Communication Networks, January 2002.

[RFB01] K. G. Ramakrishnan, S. Floyd, and D. L. Black. The addition of explicit congestion notification (ECN) to IP. RFC 3168, Internet Engineering Task Force, September 2001.

[RHE00] R. Rejaie, M. Handley, and D. Estrin. Layered quality adaptation for internet video streaming. *IEEE Journal on Selected Areas of Communications (JSAC), Special Issue on Internet QOS.*, 2000.

[SCFJ96] Henning Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: a transport protocol for real-time applications. RFC 1889, Internet Engineering Task Force, January 1996.

[Sch03] H. Schulzrinne. Internet technical resources - long-term traffic statistics. Web Page: http://www.cs.columbia.edu/~hgs/internet/traffic.html, March 2003.

[She95] S. Shenker. Fundamental design issues for the future internet. *IEEE Journal on Selected Areas in Communications*, 13:1176–1188, 1995.

[SLD02] F. Paganini S. Low and J. Doyle. Internet congestion control: An analytical perspective. *IEEE Control Systems Magazine*, February 2002.

[UCB] UCB/USC/LBNL/VINT. Network simulator ns (version 2). Avaliable at: http://www.isi.edu/nsnam/ns/.

[Unb97] H. Unbehauen. *Regelungstechnik I. Klassische Verfahren zur Analyse und Synthese linearer kontinuierlicher Regelsysteme*. Vieweg, 9 edition, 1997.

[Wel02] M. Welzl, et al. Interpretation of ECN as a less severe congestion signal. E-mail discussion on ecn-interest mailing list, July 2002.

[WP02] Z. Wang and F. Paganini. Global stability with time delay in network congestion control. In *Proceedings of the IEEE Conference on Decision and Control*, 2002.

[YL01] Q. Yin and S. Low. Convergence of REM flow control at a single link. *IEEE Communications Letters*, 5(3):119–121, March 2001.

[ZBHJ97] L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource reservation protocol (RSVP) – version 1 functional specification. RFC 2205, Internet Engineering Task Force, September 1997.

[ZHK01] S. Zimmermann, T. Hamann, and U. Killat. Dynamics of different congestion pricing strategies. In J.M. de Souza, N.L.S. da Fonseca, and E.A. de Souza e Silva, editors, *Teletraffic Engineering in the Internet Era, Proc. of the International Teletraffic Congress – ITC-17, Salvador da Bahia, Brazil*, volume 4 of *Teletraffic Science and Engineering*, pages 1187–1200. Elsevier, September 2001.

[Zim03] S. Zimmermann. Verfahren zur skalierbaren, effizienten, fairen und Dienstgüteabhängigen nutzung von Ressourcen in IP Netzen. Technical report, Dept. Communication Networks, Technical University Hamburg-Harburg, August 2003.

[ZK02] S. Zimmermann and U. Killat. Resource marking and fair rate allocation. In *Proc. of the ICC 2002, New York*, volume 2, pages 1310–1314. IEEE, April 2002.

[ZK04] S. Zimmermann and U. Killat. Control theoretic model for congestion pricing and impact of instability. *AEÜ, International Journal of Electronics and Communications*, accepted for publication, 2004.