TCP-Friendly Congestion Control for Multimedia Communication in the Internet

Vom Fachbereich 12 (Elektrotechnik)
der Technischen Universität Berlin
zur Verleihung des akademischen Grades
Doktor-Ingenieur
genehmigte Dissertation

von
Dipl.-Ing. Dorgham Sisalem

Berlin 2000

D 83

Tag der Einreichung:
13. Juni 2000

Tag der wissenschaftlichen Aussprache:
10. Oktober 2000

Promotionsauschuss:
Vorsitzender: Prof. Dr.-Ing. E. Obermeier
1. Berichter: Prof. Dr.-Ing. A. Wolisz
2. Berichter: Prof. Dr. Dr. h.c. R. Popescu-Zeletin (FB 13)

Abstract

Due to its simple design and flexible packet switching based architecture the Internet is gradually becoming the communication infrastructure for multimedia applications. While data applications such as WEB and FTP are based on TCP, multimedia applications will be based on UDP due to its flexibility and support of multi-point communication. However, as UDP does not support congestion control, wide usage of multimedia applications in the Internet might lead to overload situations and cause the starvation of congestion controlled TCP traffic which reduces its bandwidth share during overload situations. To avoid such a situation, UDP-based applications must be enhanced with congestion control algorithms that collect, exchange and process information about the network congestion status and adjust the behavior of the end systems based on this information. As evaluation criteria for the developed mechanisms we consider in this work the aspects of friendliness of UDP traffic towards competing TCP connections, the consideration of multimedia specific characteristics and the scalability of the schemes for large multicast communication. In this context, TCP-friendliness indicates that competing TCP and UDP flows with similar loss and delay values should receive on the average similar bandwidth shares. In this work, we specify, implement and investigate novel TCP-friendly congestion control schemes for UDP-based multimedia communication. The efficiency of the developed schemes is shown through simulations and measurements over the Internet as well as through comparisons to similar approaches proposed in the literature.

For the collection and exchange of information about the network load, losses and delays the here presented schemes use the real-time transport protocol (RTP) widely used for multimedia communication in the Internet. While RTP already offers the necessary mechanisms for collecting and exchanging information about losses and delays it only supports the exchange of information in intervals of a few seconds. Therefore the here presented schemes are designed to achieve an optimal adaptation behavior on the one side and take the infrequency of the RTP feedback messages on the other.

In the first part considering point-to-point communication the advantages and disadvantages of using RTP for congestion control are discussed and an end-to-end congestion control scheme called loss-delay based adaptation (LDA) is developed and investigated.

In the second part of the work, the effects of and the constraints imposed by multimedia specific characteristics on the congestion control process are investigated. Here, a general framework for incorporating user and content specific parameters with the TCP-friendly congestion control procedure is presented and evaluated.

In the third part of the work, the issue of congestion control in heterogeneous multicast environments is discussed and solutions for scalable feedback and measurements are presented. Here, a scheme called multicast enhanced loss-delay based adaptation (MLDA) is developed. MLDA adjusts the transmission rate of the sender in a TCP-friendly way in accordance with the receivers' heterogeneity.

Finally, while in the first three parts of the work the control functionalities are confined to the end systems, in the last part of the work the advantages and disadvantages of extending the network routers to inform the end systems about resource availability in the network are discussed. In this context, we investigate various schemes proposed for rate allocation discussed for the available bit rate (ABR) service proposed for ATM and present a similar yet scalable IP-based scheme called adaptive load service (ALS) to be used in the Internet.

Keywords: Internet; TCP; RTP; TCP-friendliness; Congestion control; Multimedia communication; Multicast; ABR.

Zusammenfassung

Aufgrund seiner einfachen Architektur wird das Internet zunehmend als Netzinfrastruktur für multimediale Kommunikation genutzt. Wegen seiner Einfachheit und Unterstützung von Mehrpunkt-Kommunikation verwenden multimediale Anwendungen meist das UDP-Protokoll. Da UDP keine Staukontrollmechanismen unterstützt, kann ein vermehrter Einsatz solcher Applikationen zu Netzüberlastung und Unterdrückung von staukontrolliertem TCP-Verkehr führen. Um dies zu vermeiden, müssen daher UDP-basierte Applikationen um Staukontrollalgorithmen erweitert werden, die Informationen bezüglich der Netzüberlast sammeln, austauschen und auswerten und das Verhalten der Endsysteme anhand dieser Informationen adaptieren. Diese Mechanismen müssen die spezifischen Eigenschaften der multimedialen Applikationen berücksichtigen. Ferner muß die Skalierbarkeit und "Freundlichkeit" dieser Anwendungen TCP-Verkehr gegenüber sicher gestellt werden. In diesem Kontext bedeutet TCP-Freundlichkeit, daß UDP- und TCP-Ströme, die ähnliche Verluste und Verzögerungen erleiden, ähnliche Anteile der verfügbaren Netzressourcen erhalten.

In dieser Arbeit werden neuartige Staukontrollalgorithmen für UDP-basierte multimediale Kommunikation vorgestellt, die sich TCP-Verkehr gegenüber freundlich verhalten. Die Effizienz der entwickelten Mechanismen wird anhand von Simulationen und Messungen im Internet gezeigt und mit verschiedenen in der Literatur vorgeschlagenen Ansätzen verglichen.

Zum Sammeln und Austausch von Kontroll- und Statusinformationen benutzen die hier vorgestellten Verfahren das "Real-Time Transport Protocol" (RTP), das im Rahmen von multimedialer Kommunikation im Internet bereits weitverbreitet ist. RTP erlaubt den Austausch von Kontrollinformationen nur in Intervallen von einigen Sekunden. Die hier entwickelten Algorithmen sind jedoch so optimiert, daß ein geeignetes Adaptationsverhalten trotz dieser Einschränkungen gewährleistet wird.

Im ersten Teil der Arbeit werden im Bereich Punkt-zu-Punkt-Kommunikation die Vor- und Nachteile von RTP bezüglich Staukontrolle analysiert, und ein neuer Algorithmus zur Staukontrolle namens "Loss-Delay Based Adjustment Algorithm" (LDA) wird vorgestellt und untersucht.

Im zweiten Teil der Arbeit werden die spezifischen Charakteristika multimedialer Applikationen beschrieben und deren Einflüsse auf den Staukontrollprozeß untersucht. In diesem Kontext werden multimediale Applikationen anhand eines parametrisierten Modells dargestellt. Ferner wird ein Lösungsansatz vorgestellt, der es erlaubt Randbedingungen, die vom Benutzer und der Applikation gestellt werden, mit den Anforderungen des Netzes zu integrieren.

Im dritten Teil der Arbeit wird die Staukontrollproblematik bei Mehrpunkt-Kommunikation untersucht. Es werden skalierbare Mechanismen zur Verzögerungsmessungen und zum Austausch von Kontrollinformationen vorgeschlagen und getestet. Hier wird ein Algorithmus namens "Multicast Enhanced Loss-Delay Based Adjustment" (MLDA) zur Staukontrolle für Mehrpunkt-Kommunikation vorgestellt, der die Senderate des Senders unter Berücksichtigung der Heterogenität der Empfänger und des Netzes in einer TCP-freundlichen Weise anpaßt.

In den ersten drei Teilen der Arbeit ist die Kontrollfunktionalität ausschließlich bei den Endsystemen angesiedelt. Im letzten Teil der Arbeit untersuchen wir zusätzlich die Vor- und Nachteile einer Kooperation zwischen den Endsystemen und den Netzknoten, wobei die Netzknoten den Endsystemen Informationen bezüglich der vorhandenen Kapazitäten mitteilen.

Schlagwörter: Internet; TCP; RTP; TCP-Freundlichkeit; Staukontrolle; Multimedia; Multicast; ABR.

Acknowledgments

First of all I would like to express my deepest appreciation, gratitude and thanks to my advisor Prof. Dr.-Ing. Adam Wolisz for his guidance and support as well as inspiring discussions and constructive criticism. His academic excellence and farsightedness besides his sincerity and straightness were crucial ingredients to my work at the different stages of its development.

Further, I would like to express my gratitude to Prof. Dr. Dr. h.c. Radu Popescu-Zeletin for taking the time to evaluate my work and for his invaluable comments. I would also like to thank him for the excellent research environment he created at the GMD FOKUS that made this work possible in the first place.

My special thanks go to Prof. Henning Schulzrinne, not only for his advise and cooperation throughout the past years but also for his trust and understanding. His work, attitude and way of thinking were more than inspiring to me.

Additionally, I am indebted to Prof. Mikhail Smirnov for giving me the freedom to combine my doctoral work with my work as a researcher at the Global Networking (GloNe) group at the GMD FOKUS. I am also grateful for our long discussions and his ability to always show me the positive side of things.

I would also like to thank Dr.-Ing. Henning Sanneck who started and finished his PhD work at the GMD FOKUS at the same time as I did. Working next to such an excellent researcher was not only a pleasure but a constant incentive to go a step further with my own work as well.

I express my worm thanks to the entire GloNe group especially Dr.-Ing. George Carle, Ilona Schubert, Rudolf Roth and Laurensius Tionardi for their permanent support and encouragement as well as discussions and proof-reading of my work.

Many thanks go to Martin Reisslein, Andrey Vasilyev and the ICSI system administrators for providing the measurement end points, to Timur Friedman and Frank Emanuel for implementing the basic RTP model and to Matthias Kranz for supporting me with the delay measurements.

My heartiest gratitude go finally to my entire family, to my parents, Awni and Luzie, who always provided me with a loving and comforting home withstanding the political and social turbulence around us and always supported and encouraged my studies, to my brother Heider and sister Rima for their invaluable friendship and for always being there when needed, to my daughter Juliana for being so sweet and loving and last but not least to my wife Marta for her loving and understanding and for her encouragement and support throughout the past years.

Contents

1 Introduction **1**
1.1 Motivation . . . 1
1.2 Goals and Overview . . . 5

2 Architecture, Characterization and Modeling of the Internet **9**
2.1 Basic Architectural Overview of the Internet . . . 9
2.1.1 The Internet Layer . . . 10
2.1.2 The Transport Layer . . . 13
2.1.3 The Application Layer and Application Related Control . . . 18
2.1.4 The Host-Network Interface . . . 22
2.2 End-To-End Estimation of Path Characteristics in the Internet . . . 22
2.2.1 Estimation of Round Trip Time . . . 22
2.2.2 Estimation of Losses . . . 23
2.2.3 Estimation of Bottleneck Bandwidth . . . 24
2.3 Investigation Environment . . . 25
2.3.1 Simulation Models and Environment . . . 26

3 Traffic Management in the Internet **31**
3.1 Congestion Control in the Internet . . . 31
3.1.1 Evaluation Criteria for Congestion Control Mechanisms . . . 34
3.1.2 End-to-End Unicast Congestion Control Schemes . . . 38
3.1.3 Congestion Control for Multicast Communication . . . 41
3.1.4 Network-Based Adaptation . . . 49
3.2 Quality of Service Control for Multimedia Communication . . . 51
3.2.1 The Integrated Services Model (IntServ) . . . 51
3.2.2 The Differentiated Services Model (DiffServ) . . . 52
3.2.3 QoS Architectures . . . 53
3.3 Adaptation versus Reservation . . . 54

4 TCP-Friendly Congestion Control for Point-to-Point Communication **57**
4.1 Implications of the Feedback Mechanism on the Design of the Congestion Control Scheme . . . 57
4.2 Pitfalls of RTP-Based Adaptation Schemes . . . 58
4.3 The Loss-Delay Based Adjustment Algorithm (LDA) . . . 60
4.3.1 Measurement of Path Characteristics . . . 60

4.3.2 Investigation of Design Options for Specifying Adaptation Schemes 61
4.3.3 Rate Adjustment with LDA 73
4.4 Performance of the Loss-Delay Based Adjustment Algorithm 74
4.4.1 Fairness and Convergence of the LDA Algorithm 75
4.4.2 Investigation of the TCP-Friendliness of LDA 83
4.5 Measuring the Performance of LDA over the Internet 90
4.6 Comparison of the Performance of LDA to Other Congestion control Schemes 95
4.6.1 Comparison of LDA with a Rate-Based TCP-Friendly Adaptation Scheme 95
4.6.2 Comparison of LDA with an Equation-Based TCP-Friendly Adaptation Scheme 96
4.7 Summary 97

5 Constrained Congestion Control for Multimedia Communication 99
5.1 A Generic Model for Multimedia Communication 99
5.2 A General Framework for Rate Adaptation for Constrained Multimedia Flows 101
5.3 Performance Investigation of CTFAF 104
5.3.1 Effects of the Length of the Adaptation Steps ($S_{\text{adaptation}}$) on the Performance of CTFAF 105
5.3.2 Effects of the Length of the Resetting Period (T_{reset}) on the Performance of CTFAF 107
5.3.3 Effects of Delays and Flow Numbers on the Performance of CTFAF 107
5.4 Summary 110

6 TCP-Friendly Congestion Control for Multicast Communication 111
6.1 Multicast Enhancement for LDA 112
6.2 Multicast Enhanced Loss-Delay Based Adaptation Framework 115
6.2.1 Sender Behavior 117
6.2.2 Receiver Behavior 118
6.2.3 Scalable Feedback 119
6.2.4 Receiver-Based Measurement of Path Characteristics 120
6.2.5 Synchronous Join and Leave Actions 124
6.2.6 Reliability Issues 125
6.2.7 Parameter Settings 125
6.3 Integration of RTP and MLDA 126
6.4 Performance Investigation of MLDA 127
6.4.1 Simulation Model of MLDA 127
6.4.2 Performance of MLDA in Heterogeneous Multicast Environments . 128
6.4.3 Performance of MLDA in Dynamical Environments 130
6.4.4 Performance of MLDA in Multicast Environments with Shared Links 131
6.4.5 Scalability Performance of MLDA 132
6.5 Comparison of the Performance of MLDA to Other Congestion Control Schemes 133
6.5.1 Comparison of MLDA and PLM 134

6.5.2 Comparison of MLDA and RLC 136
6.5.3 Comparison of MLDA and IRFMC 138
6.6 Summary 139

7 Network-Assisted Congestion Control 141
7.1 Motivation 142
7.2 Specification of the Adaptive Load Service (ALS) 144
7.2.1 Sender Behavior 144
7.2.2 Router Behavior 145
7.2.3 Receiver Behavior 147
7.3 Implementation Issues 149
7.3.1 Distribution of Control Information 149
7.3.2 Processing Overload of RTCP Messages 150
7.3.3 Parameter Settings 150
7.4 Performance of the Adaptive Load Service 150
7.4.1 Competing TCP and ALS Traffic 151
7.4.2 Performance of ALS in Terms of Max-Min Fairness 155
7.4.3 Performance of ALS in Heterogeneous Environments 156
7.5 Summary 158

8 Summary and Future Work 159

A Investigation of ABR Switch Mechanisms 163
A.1 Introduction 163
A.2 Congestion Control and Avoidance in ATM Networks 164
A.2.1 The MIT Scheme 164
A.2.2 The OSU Congestion Avoidance Scheme 165
A.2.3 Congestion Avoidance using Proportional Control (CAPC) 165
A.3 Simulation Environment 166
A.4 Fairness Testing Criterion 166
A.5 Performance Results 167
A.5.1 The OSU Scheme 168
A.5.2 The CAPC Scheme 170
A.5.3 The MIT Scheme 172
A.6 Summary 174

Chapter 1

Introduction

Due to its simple design and flexible packet switching based architecture the Internet [179] has become the standard network for data communication and is continuously gaining in popularity. Actually, with the amount of data carried over the Internet doubling each year [36], various studies [186, 1] predict that the volume of data traffic will in a few years time overtake voice traffic.

In addition to traditional data communication such as Email or data retrieval, the ease of transmitting data packets between two points and the possibility of distributing data to multiple end systems using multicast [39] has encouraged the usage of multimedia communication over the Internet [47] and lead to various activities regarding telephony [64, 155, 159] and video conferencing [174, 156, 45].

Deploying transport protocols that support congestion control mechanisms for data transfer played a major role in avoiding overload situations and thus stabilizing the Internet. To maintain this stability, similar congestion control mechanisms need to be introduced with the deployment of multimedia communication in the Internet. With such congestion avoidance mechanisms, data losses resulting from possible overload situations are also reduced. This in its turn results in an improved quality of service for the multimedia flows.

1.1 Motivation

The rapid growth, increasing bandwidth and the availability of low-cost multimedia end systems has made it possible to use the Internet for multimedia applications ranging from telephony to conferencing, distance learning, media-on-demand and broadcast applications [157]. However, the Internet resembles to some extent a highway. Everybody can drive in it but the roads only have a limited capacity. This leads to congested roads, long waiting periods and even accidents. To avoid this situation one might increase the capacity of the roads by adding new lanes. However, this is a rather expensive solution. A less expensive solution for providing at least a subset of the cars such as police cars or ambulances with a better service is by giving these cars a higher priority than others. This can be realized for example, by mandating that all cars with lower priority should move aside when a higher priority car expresses its need to go through by signaling a blue or red flashing light. Another method would be to dedicate special lanes for higher priority cars and prohibit normal cars from using these lanes. For normal drivers, i.e., drivers not allowed to use the dedicated lanes or blue lights the only way to reduce road congestion and thus reduce the possibility of building up traffic jams is by adapting to the road situation. For example, a few drivers might join in a car-pool and drive together and hence reduce the number of cars entering the road and thereby reduce the possibility of road congestion. Another approach would be to refer to a transportation method that provides lower personal comfort such as trains or buses or even delay the trip to a time with lower road congestion. So, while the drivers now do not end up getting the service they wanted, i.e., to drive with their cars over congestion free roads from one point to the other, by shifting to a transportation method with possibly lower comfort such as buses or car-pools they contribute to a reduction of the road congestion and hence increase the possibility that their alternative transportation vehicle gets to its destination with lower congestion delays.

In a similar manner, the Internet does not in general limit the number of data packets sent by the end systems. That is, the only limitation on the transmission rate of some system is its own performance and the capacity of the link connecting it to the Internet. The major resource to consider in a network is its bandwidth, i.e., in the context of this study the number of packets it can accept and forward during a time interval. Thus, by sending data packets without any consideration of the available bandwidth, overload situations due to congestion are unavoidable. Note, however, that while for the case of the highways, congestion situations merely result in longer delays, for the case of the Internet some data packets are delayed while others get discarded all together. This would then lead to data communication with high losses and long delays. Similar to a highway, the overload problems can be avoided by increasing the available bandwidth. Alternatively, for a subset of the users a higher quality can also be supported by dedicating a part of the bandwidth to those users. However, both approaches of either increasing the amount of available resources or dedicating a part of the available resources to a subset of the users require substantial changes to the Internet and can thus only be realized at a high cost and over long transition periods. In order to avoid network congestion and improve the quality of service perceived by all users, the users need to adapt their transmission and reception behavior in accordance with the available network resources. Hence, while for the case of the highway, mechanisms are used that reduce the number of cars entering the highway, for the case of the Internet the end systems should deploy mechanisms that adjust the number of packets entering the network in accordance with the available resource.

Severe and consistent congestion states might not only result in high losses and large delays but also lead to congestion collapse [53]. Such a situation occurs when the network is utilized to a very high degree but with only a small amount of useful data reaching the receivers. Congestion collapse in the Internet was first reported by Nagle in [121], and was largely due to the version of the data transport protocol called TCP used then. TCP achieves reliable data transport by retransmitting packets for which no acknowledgment were received after a specific time. Due to the severe congestion, the TCP connections unnecessarily retransmitted data packets that were either in transit or had already arrived at the receivers. This form of collapse has since been avoided by enhancing TCP to adapt the transmission behavior of the senders in accordance with the network congestion state.

That is, while no losses are indicated the sender can increase its transmission rate whereas after a loss notification the transmission rate is considerably decreased [75].

While congestion controlled TCP connections carrying time insensitive FTP or WWW traffic still constitute the major share of the Internet traffic today [185], recently proposed real-time multimedia services such as IP-telephony and group communication will be based on the user datagram protocol (UDP) [37]. While UDP does not offer any reliability or support congestion control mechanisms, it has the advantage of not adding delays to the carried data due to retransmissions as is the case with TCP. Additionally, as UDP does not require the receivers to send acknowledgments for received data it is well suited for group communication. However, UDP's lack of congestion control mechanisms might lead to another form of congestion collapse, that arises when network resources are wasted by transporting packets through the network that are dropped before reaching the receiving end system. This situation arises when the sender is transmitting data at a rate higher than the rate of some bottleneck on the path towards the receiver. Additionally, deploying UDP in the Internet on a large scale might result in extreme unfairness towards competing TCP traffic. In response to losses in the network, TCP connections sharing the same congested links with UDP flows reduce their transmission rates. However, without any rate reduction on behalf of the non-congestion-controlled traffic, the TCP connections would starve and receive a much smaller bandwidth share than the competing UDP flows. Therefore, UDP flows need to be enhanced with control mechanisms that adapt the transmission and reception behavior of the end systems in accordance with the available resources. Such adaptation schemes should not only aim at avoiding network overload but be also fair towards competing TCP connections, i.e, be *TCP-friendly*. TCP-friendliness indicates here, that if a TCP connection and an adaptive flow with similar transmission behaviors have similar round trip delays and losses both flows should receive similar bandwidth shares. While such adaptation schemes do not guarantee any QoS levels, they indirectly improve the perceived quality by the users by avoiding congestion. This leads to reduced losses on the one side and increased overall utilization on the other. It is usually the case that due to the loss of content or the need to use a considerable overhead for forward error correction a low bandwidth video stream, for example, with no or only low losses can have a higher perceived quality than a high bandwidth yet lossy stream [189].

For the case of point-to-point communication, by deploying adaptation mechanisms end systems can benefit from any available resources in the network on the one hand and avoid overloading the network on the other. Based on feedback information from the receiver or the network, the sender can adjust its transmission level in accordance with the network congestion state as observed by the receiver or the network nodes. Such an approach, where the sender actively changes the amount of data injected into the network is called *sender-based* adaptation.

For the case of group communication, the current Internet supports the so called multicast [179] paradigm. With multicast the sender addresses its data to an address that represents a multicast session and not actual physical receivers. A receiver wishing to receive the multicasted data needs to inform the network routers about this. Only in this case do the routers forward the data towards the receiver. Using a multicast routing protocol [192], the network establishes a routing tree from the sender to the receivers.

New receivers wishing to join the multicast session do not cause the establishment of separate multicast routing trees. Instead, the multicast tree is enlarged to include the new receivers. The sender needs only to transmit one stream of data, that is distributed by the network routers to the receivers. This approach reduces the amount of data that has to be transported in the network as the data is only replicated at certain division points in the multicast tree. However, due to the heterogeneity of the traversed paths to the receivers and the different capabilities of the receivers, a single sender transmission rate cannot satisfy the conflicting bandwidth requirements at different sites. Therefore, with sender-based adaptation schemes the sender rate is usually adapted to the requirements of the worst positioned receiver, thereby reducing the quality of the data perceived by all receivers.

To avoid the worst receiver problem in the sender-based adaptation schemes, various proposals have been made for layered data distribution. Those proposals are often based on partitioning a data stream into a base layer containing the information needed to achieve the lowest quality representation and a number of enhancement layers [191, 113, 200]. The different layers are then sent on different multicast sessions and the receivers determine how many sessions to receive and thereby adjust their QoS in respect to their own requirements and capacities. To avoid the need to resynchronize different levels at the receivers back into one continuous stream, other schemes propose to send the same data content [104, 95] in parallel on different sessions each with different QoS parameters. The receivers need then only to receive one stream that is appropriate for their capacities. With layered transmission the congestion avoidance is achieved using the multicast capabilities of the network. Data is only forwarded over some link, if a receiver connected to the multicast distribution tree through this link wants the data. If the receiver leaves a layer, then the routers will not forward the data of this layer to the receiver and the amount of data packets traversing the path from the sender to the receiver will hence be reduced.

While layered data transmission schemes, also noted as *receiver-based* adaptation mechanisms, solve the heterogeneity problems, they might cause additional delays at the receivers and increase the complexity of the end systems. Additionally, current schemes usually set the size of the sent layers statically without regard to the actual heterogeneity of the network and receivers. To better account for the different capabilities of the receivers and the dynamically changing network congestion state the senders should adjust the sizes and numbers of layers they transmit in accordance with these parameters.

Both sender- and receiver-based adaptation schemes share one major problem of having to probe the network in order to determine the appropriate amount of resources to use. For the sender-based schemes, the sender needs to gradually increase its transmission rate until congestion is observed. This problem is amplified in the case of receiver-based schemes. Usually, only a small number of data layers of greatly varying bandwidth consumption are used. Thus, joining a higher layer usually introduces a sudden and large load to the network which might lead to network overload and high losses. To avoid this probing problem it would be beneficial if the senders and receivers could base their adaptation behavior on more detailed information about the available resources in the network than the information collected at the end systems about the losses and delays. This would, however, require the involvement of network nodes in the adaptation procedure to provide such information.

1.2 Goals and Overview

When using TCP for a file transfer the time required for transferring the file between two end systems is usually the major consideration. For the case of adapting UDP-based multimedia traffic several other aspects need to be considered as well:

- Congestion avoidance,
- improve the quality of service by reducing the losses and achieving the highest possible bandwidth share,
- avoid starvation of competing TCP traffic and
- support group communication with a large number of heterogeneous end points.
- Additionally, there are usually minimum and maximum limits on the bandwidth requirements of a multimedia flow below which the content is no longer understandable and above which no perceivable improvement in the quality is noticed. Further, frequent and large changes in the amount of data arriving at the receivers might result in a rapidly changing perceived quality which can be rather annoying to the user. Thus, adaptation schemes need to aim at achieving a stable level of transmission and reception rates at the end systems.

The main goal of this work, will be to specify and investigate TCP-friendly congestion control schemes for UDP-based multimedia communication. In this context, we will be addressing both point-to-point as well as group communication in the Internet. The basic approach here for realizing effective congestion control is by collecting information about the congestion state of the network and adjusting the transmission rate of the senders and reception behavior of the receivers in accordance with the network congestion state. The adaptation of the behavior of the end systems is to be realized in a manner as to consider the different aspects of congestion avoidance, fairness and stability. To avoid introducing a new protocol for collecting state information we rely in this work on the real-time transport protocol (RTP) [158]. RTP was standardized by the IETF and is widely used for multimedia communication in the Internet. RTP supports the exchange of end-to-end feedback information describing losses, delays and other information collected at the end systems. In contrast to various other protocols used for adaptation mechanisms that provide frequent feedback information on the scale of one or a few round trip times [195, 91] RTP provides only infrequent information on the scale of a few seconds. This infrequency of the feedback not only imposes novel challenges on the design of adaptation algorithms but also limits the achievable benefits gained using adaptive end systems as the end systems can not adjust their behavior to benefit fast enough from rapid changes in the network conditions. Thus, the goal here will be to adjust the end systems behavior to the average available bandwidth and not react to rapid changes in buffer lengths of the routers for example. Actually, in the context of multimedia communication this might be more appropriate than rapidly changing the transmission rate at a high frequency as this would usually result in annoying variation in the quality.

To cover the different aspects of TCP-friendly congestion control for unicast as well as multicast communication and give a general overview of related work to these problems the work will be structured in the following chapters:

Architecture, Characterization and Modeling of the Internet:
This chapter gives a short introduction to the Internet architecture, the protocols for realizing multimedia group communication and our simulation environment. Additionally, we describe some of the available approaches for estimating the characteristics of Internet paths.

Traffic Management in the Internet:
This chapter gives a brief overview of various schemes for congestion control for unicast and multicast communication in the Internet. In this context, we additionally present different classification possibilities as well as evaluation aspects of congestion control schemes. Further, different approaches for supporting guaranteed QoS in the Internet as well and their advantages and disadvantages compared to congestion control schemes are shortly discussed.

TCP-Friendly Congestion Control for Point-to-Point Communication:
The basic approach here for realizing congestion control is for the receiver to collect information about the network congestion state and inform the sender about the estimated state. The sender can then increase its transmission rate in case of no congestion, otherwise it needs to reduce it. In this chapter, we first describe some of the restrictions and benefits of using RTP with its infrequent feedback messages for establishing the control loop. We further present some of the effects of designing adaptation schemes without taking the aspect of TCP-friendliness into account. As an example for a TCP-friendly congestion control we present in this chapter an algorithm called the loss-delay based adaptation algorithm (LDA) [173]. Based on end-to-end measurements of losses, bottleneck bandwidth and round trip time LDA determines the appropriate rate to use. To identify appropriate methods for increasing and decreasing the rate with LDA we investigate some of the available mechanisms in the literature for adapting the transmission behavior of senders based on the feedback information of the receivers [166] and look at their performance in terms of stability and TCP-friendliness. Due to the restrictions of most of the investigated schemes we present novel ways to be used with LDA for increasing and decreasing the transmission rate in a flexible and TCP-friendly manner. The efficiency of LDA and its performance under a wide range of parameters is then investigated using simulations and measurements as well as comparisons to some of the most recently proposed TCP-friendly adaptation algorithms.

Constrained Congestion Control for Multimedia Communication:
Congestion control schemes generally assume that the sender can adjust its transmission rate in accordance with the values determined by the adaptation scheme and in accordance with the dynamics of the TCP-friendly algorithm with arbitrary granularity and with no boundaries as to the maximum or minimum bandwidth

values to use. However, in reality there are certain maximum and minimum bandwidth requirements for multimedia contents above and below which the sent data is meaningless for the receiver. Additionally, the used compression style for a multimedia content might dictate that the changes in the transmission rate of a video stream for example can only be realized in a granular way. Also, there might be some restrictions arising from the user himself with regard to the acceptable variations in the perceived QoS over a time interval. Such restrictions depend primarily on the transfered content, the used compression style, the communication scenario and the user himself. To investigate the effects of such restrictions on the adaptation process we present in this part of the work a generic model of a multimedia source and describe some constraints such a model might impose on the adaptation process. Based on this model we further describe a general framework to assist rate adaptation algorithms in incorporating various constraints in the adaptation process on the one hand and retain the fair bandwidth distribution characteristics required of the algorithms on the other.

TCP-Friendly Congestion Control for Multicast Communication:
In this chapter, we describe a general TCP-friendly congestion control approach for handling heterogeneous multicast communication called multicast enhanced loss-delay based adaptation algorithm (MLDA) [178]. MLDA is a hybrid sender and receiver-based adaptation scheme that combines on the one hand various well known concepts for multicast congestion control such as receiver-based rate calculation presented by Handley et al. [61] and layered transmission [113] into a unified congestion control architecture. On the other hand, MLDA provides novel solutions for dynamic layering [167], round trip delay measurements, scalable feedback collection and a general framework for the cooperation between the sender and receivers that enables a seamless integration of those concepts with the concept of layered transmission and allows for efficient congestion control. With MLDA the receivers use LDA to estimate the appropriate TCP-friendly bandwidth share on the path between the sender and them. The sender collects those estimations and shapes the number and size of the transmitted data layers based on those values. The information about the number and sizes of the layers is then included in the sender's reports. Based on this information and their own estimation of bandwidth availability the receivers can determine the appropriate number of data layers to join. Similar to the case of point-to-point communication, RTP is used here as well for collecting and distributing control information. We, additionally, look at some extensions for RTP for improving its scalability on the one hand and increasing the rate at which feedback messages are sent on the other. As all receivers need to estimate the round trip delay between them and the sender we present and test a scalable measurement approach for the delay estimation that introduces minimal control overhead. Finally, the performance of MLDA is investigated using simulations and compared to that of a number of recently proposed TCP-friendly congestion control schemes for multicast communication.

Network-Assisted Congestion Control:

On the one hand, sender-based adaptation mechanisms used for the unicast communication can not accommodate heterogeneous receivers adequately. On the other hand, receiver-based approaches as were discussed for multicast communication, introduce considerable complexity to the end systems. As a more general solution we propose a congestion control approach called the available load service (ALS) [175] that benefits from both approaches and utilizes information supplied by the network nodes to optimize the adaptation decision. ALS was designed in a similar fashion to the available bit rate service (ABR) [162] proposed for the asynchronous transfer mode (ATM) [127] networks. That is, the senders transmit control messages indicating their requirements, intermediate routers adjust the indicated values in accordance with their available resources and the receivers send the content of the control messages back to the senders. The transmission behavior of the senders is then adjusted in accordance with the indicated values in the returned control messages. Based on the information about the available bandwidth on different parts in the network, the sender can effectively distribute its data stream on different layers. By including information about the data distribution in the sender's control messages the receiver can determine the appropriate data layers to receive and hence avoids the need for probing the network and the complexity attached to this procedure. Unlike ABR, ALS is IP-based and was designed to scale for large multicast groups and accommodate the needs of heterogeneous receivers.

Summary and Future Work:

This chapter gives a brief summary of the results of the work and we look at some of the open issues and future research topics.

Chapter 2

Architecture, Characterization and Modeling of the Internet

The focus of this work will be on realizing TCP-friendly congestion control schemes for multimedia communication over the Internet using adaptive control mechanisms. While a general overview of congestion control and traffic management in the Internet will be presented in chapter 3, this chapter gives a brief description of the basic architectural concepts of the Internet in Sec. 2.1 and provides a general introduction to the relevant protocols in this context. In Sec. 2.2, we present some of the problems of estimating the characteristics of Internet links and possible approaches for solving these problems. Finally, Sec. 2.3 describes the simulation environment used throughout this study for investigating different adaptation approaches.

2.1 Basic Architectural Overview of the Internet

Unlike traditional circuit-switched telecommunication networks the Internet is based on packet-switching. That is, in a circuit-switched network each data flow receives dedicated bandwidth which emulates a dedicated physical line between the sender and the receiver of the flow. On the contrary, in a packet-switched network multiple data flows are multiplexed on the same line with all of the flows competing for the available resources.

As depicted in Fig. 2.1, in the basic communication scenario in the Internet we can distinguish between end systems and network nodes. In this scenario, a flow of data is generated from one end system and sent to another over a data path traversing one or more network nodes. In short, an end system consists of an application that generates content or data. This data is divided into separate data chunks called packets and addressed to a specific application at another end system using the transport layer. The internet layer is responsible for routing and moving the data packets around in the network so that they actually arrive at the other end system. At the network nodes, incoming data packets are buffered, processed and forwarded to either another network node or to an end system. Note, that not every end system or network node needs to have all the components depicted in Fig. 2.1. Thus, some networking devices consist only of a physical device that simply copies all incoming packets to all of its outputs and does not use an internet layer.

Further, in some cases the networking devices need to process the transport or application layer headers for security reasons [11] or in order to add novel services [7].

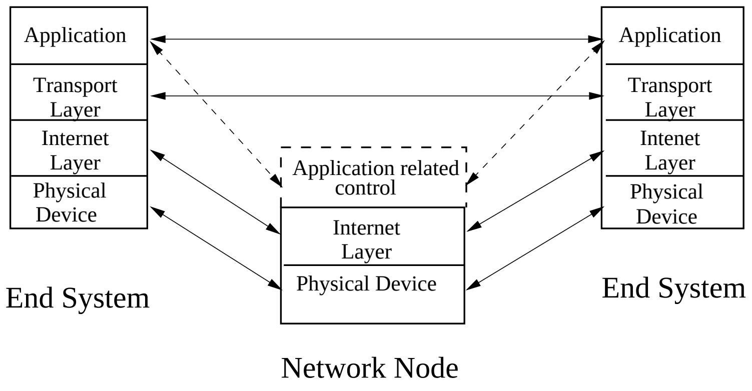


Figure 2.1: General communication scenario in the Internet

In this section, we give a brief introduction to the general protocol architecture of the Internet, usually called the TCP/IP protocol suite, and present some of the issues related to our work.

2.1.1 The Internet Layer

The internet layer handles the routing of data packets from their origin to their destination in a connection-less manner. That is, no explicit initiation or termination of connections is required for carrying data between two end points. Hence, the route taken by the data packets is determined independently for each packet and packets belonging to the same data flow between a sender and a receiver might actually follow different routes.

The Internet protocol (IP) [142] is the default protocol for this layer in the Internet. IP adds a header to the actual data to be sent including mainly the addresses of the sender and receiver end systems as well as some additional fields.

2.1.1.1 Quality of Service (QoS) Control and Buffer Management in the Internet

Currently, the internet layer does not provide for any quality of service guarantees. That is, sent data packets might be arbitrarily delayed in the network and might never reach their destination. Additionally, packets may even arrive in a different order than they were sent. While the issue of packet reordering results from the connection-less nature of the internet layer, the loss and delay aspects can be attributed to the physical links over which the data is carried and more importantly the buffer handling mechanisms at the network nodes.

Network nodes in the Internet have the major task of routing traffic from an incoming link to an outgoing one. To absorb the burstiness of the incoming traffic and forward it at the speed of the outgoing link routers deploy buffers in which the incoming packets are queued in and then read from for forwarding. Out of the numerous proposals for buffer management schemes we just pick a few that are either widely used or are being considered for wide deployment.

First In First Out (FIFO): This is the simplest form of buffer management and is, due to its simplicity, the most widely used approach in the Internet routers. Packets queued into a FIFO buffer are played out at the same sequence they were queued in. Arriving packets at a router are discarded whenever the buffer reaches its limit. While this scheme is trivial to implement, it can lead to synchronization problems and unequal bandwidth distribution among competing streams. The synchronization problem observed for TCP connections [54] arises in the situation when an overloaded router discards packets arriving from multiple connections. Due to the congestion control mechanisms all the connections reduce their transmission behavior more or less at the same time which results in emptying the overloaded buffer. Thus, all the connections start increasing their transmission window again until the router's buffer is overloaded again and the cycle is repeated. Such a behavior naturally results in a poor network performance as the network stays underutilized during the window increase phase.

In addition to the synchronization problem, FIFO buffers have severe fairness problems. As the drop probability for all competing connections is equal, a greedy connection utilizing a large bandwidth share has the same loss percentage as a connection with a much smaller share. Thus, even though the greedy connection will most probably be more responsible for the overload situation at the router it will still receive a higher bandwidth share.

Random Early Detection (RED): A RED [55] gateway detects incipient congestion by computing the average queue size. When the average queue size exceeds a preset minimum threshold the router drops each incoming packet with some probability. Exceeding a second maximum threshold leads to dropping all arriving packets. This approach not only keeps the average queue length low but avoids synchronization effects as well. While some of the newer routers already support the RED buffer management, RED is still in general not widely deployed.

RED shows the same fairness problem as FIFO. Using a somewhat more complicated design Lin et al. [105] propose a modification to the behavior of RED to isolate greedy traffic and better protect connections that adapt their transmission rate to the network congestion state.

Fair queuing mechanism: Buffer management for the Internet has been a popular research topic since the introduction of the Internet. The developed schemes vary greatly in their complexity, scope and usability. While some approaches aim mainly at achieving an equal bandwidth distribution and isolating greedy traffic [180] other schemes such as round robin [2] or weighted fair queuing [132] aim at guaranteeing each connection a specific bandwidth share as well as a delay limit. However, most of these schemes are too complicated to use and have thus not found wide usage.

2.1.1.2 Data Distribution in the Internet

With the internet layer data packets can be distributed in one of three modes:

Unicast: Data packets are sent from one source to one specific end system. In this case, the network routers use a unicast routing protocol [66, 108, 119] for establishing a route between the sender and the receiver.

Broadcast: Data packets are sent to every other host. This is usually used in a local area network for communicating with all other hosts on a part of the network.

Multicast: Data packets are sent to a specific group of end systems.

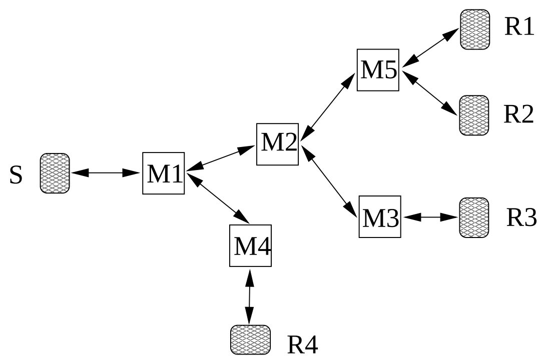


Figure 2.2: Example of multicast transmission

Group communication is supported in the Internet based on the multicast paradigm [42]. This is realized by assigning a multicast address to the collection of receivers and senders of a multicast group [40]. Senders simply use this address as the destination address of the data packets and the receivers need to inform the network about their willingness to join or leave a multicast group using a group membership protocol (IGMP) [39].

Fig. 2.2 depicts a multicast distribution tree. The sender (S) transmits its data to the multicast address (D). Without any receivers listening to session D, the data is only sent to the first multicast router ($M1$) and is not forwarded further. If a receiver ($R1$) wants to receive the data sent by sender (S) over the multicast group (D), it sends an IGMP join request for this address to its closest multicast router ($M5$). $M5$ sends then a join request to a second router ($M2$) asking for joining session D. $M2$ in its turn sends a join message to $M1$. This completes the path between the sender and $R1$ and $R1$ can now receive the data of session D. If $R2$ wants also to join the session, it sends a join message to $M5$, which also represents its closest multicast router. In this case, $M5$ forwards the data on the path connecting it to $R2$ as well. If $R1$ wants to leave session D it sends a leave request to $M5$. As $R2$ still wants to listen to session D, $M5$ only stops forwarding the data on the link towards $R1$. If $R2$ also sends a leave request, $M5$ sends a prune message to $M2$ so that $M2$ stops forwarding the data of session D to $M5$. $M2$ only sends a prune message to $M1$ if all receivers to which it forwards data have signaled their wish to leave the session. That is if $R3$ has joined session D, $M2$ does not send a prune message to $M1$ until $R3$ has left as well,

The routing tree from the receivers to the senders is established using a routing protocol in the form of an extension to existing unicast routing protocols [192, 48, 120].

As still not all routers in the Internet are capable of supporting a multicast routing protocol, multicast communication is often realized using a virtual network called the multicast backbone (MBone) [96]. MBone is called a virtual network because it shares the same physical media such as wires and routers as the Internet. The MBone allows multicast packets to travel through routers that support only unicast communication using an approach called tunneling. This is achieved by putting multicast packets in regular unicast packets at one multicast router and sending them as unicast packets to the next multicast router. There, the multicast packets are taken out of the unicast packets and either forwarded as native multicast packets or are tunneled again over a further unicast connection.

2.1.1.3 Conceptual Model of the Internet Layer

As a simplification of the internet layer we will be considering in this work an idealized network with stable capacity, propagation delays and routes. That is, all packets belonging to a flow will traverse the same path. Measurements over the Internet [137] show that this is the general case and our simplification is thus acceptable. Additionally, following common assumption about the wired Internet we will consider the loss of data packets as a congestion indication and not due to the damage of a part of the packets which can result from a lossy physical link such as a wireless one. Work done on deploying TCP over wireless links [9] shows that this is a non-trivial problem and requires special handling in the design of congestion control schemes, see also [181] for work on rate adaptation for mobile hosts. Therefore, we will thus restrict our work to wired communication.

2.1.2 The Transport Layer

The transport layer further refines the addressing of the data packets and enables the binding of a data flow not only to an end system but to a specific process or an application at the end system. In the TCP/IP protocol suite there are two vastly different transport protocols: the connection-oriented transmission control protocol (TCP) [143] that provides reliable and in-sequence data transport and the connection-less and unreliable user datagram protocol (UDP) [141].

2.1.2.1 The Transmission Control Protocol (TCP)

As already mentioned, due to the connection-less nature of the internet layer, data packets might get lost or arrive out of order at the receiver. Through the retransmission of lost packets and the resequencing of out-of-order packets, TCP provides a reliable and in-sequence flow of data packets between two hosts. TCP is concerned with things such as dividing the data passed to it from the application into appropriate sized chunks for the internet layer below, acknowledging received packets, setting timeouts for detecting packet losses, retransmitting lost packets and so on.

The functionalities of TCP can be roughly divided into the following parts:

Connection establishment and termination: In the context of TCP, the term connection-oriented means that two applications using TCP need to establish a TCP connection with each other before they can exchange data. That is, before starting the data transmission the application sends a synchronization message (SYN) to the receiving side which responds with its own SYN message. The sending application must acknowledge the received SYN message before it is allowed to transmit data to the receiver. To terminate a connection the end points need to exchange FIN messages.

Reliability: To ensure reliable communication, the TCP receiver acknowledges the reception of all arriving data packets. The TCP sender keeps a local copy of each sent packet until receiving an acknowledgment from the receiver indicating the correct reception of this packet at the receiver side. If a packet was not acknowledged after some time period t_{out}, then the packet is considered to be lost and is retransmitted. t_{out} must be set to an appropriate value that allows for fast discovery of lost packets but avoids retransmitting packets that are still on their way to the receiver. Jacobson describes in [75] an algorithm for estimating the round trip time using the average and the mean deviation of the measured round trip time samples as follows:

$$\begin{aligned} t_{\text{diff}} &= t_{\text{measured}} - t_{\text{RTT}} && (2.1)\\ t_{\text{RTT}} &= t_{\text{RTT}} + \delta \times t_{\text{diff}} && (2.2)\\ \sigma_n &= \sigma_{\text{n-1}} + \rho\left(|t_{\text{diff}}| - \sigma_{\text{n-1}}\right) && (2.3)\\ t_{\text{out}} &= t_{\text{RTT}} + \eta \times \sigma_{\text{n}} && (2.4) \end{aligned}$$

where t_{measured} is the measured round trip delay, t_{RTT} is the smoothed round trip delay, σ_{n} is the smoothed mean deviation and t_{out} is the estimated timeout value after which a TCP sender would retransmit an unacknowledged packet. The round trip delay (t_{measured}) is estimated as the time difference between sending a data packet and receiving an acknowledgment for it. The gain δ is a fraction between 0 and 1 that controls how quickly the new sample affects the smoothed round trip delay and is usually set to 0.125. ρ controls how quickly a new sample affects the mean deviation (σ) and is usually set to 0.25. η controls how much the deviation affects the timeout estimation. η was originally set to 2 and then increased to 4 in the Reno-TCP version.

In addition to the timeout based loss detection, the loss of a packet can be detected with the *fast retransmission* scheme. With fast retransmission, a packet X_{n} is considered to be lost if a source receives a number of duplicate acknowledgments, usually three, for packet $X_{\text{n-1}}$. Upon that, the lost X_{n} packet is retransmitted.

Congestion and flow control: Next to its reliability, TCP's ability to adjust its performance in accordance with the available network resources were the major reasons for its success and wide usage. TCP relies on the sliding window mechanism for sending data packets and avoiding overloading the receiving end systems. With this flow control mechanism, the TCP instance at the sending end system keeps a transmission buffer, called transmission window, in which it writes a copy of each sent packet. While there is still space left in the buffer the sending application can transmit additional packets otherwise the TCP instance refuses to accept new packets from the application and blocks the transmission actions of the application. The TCP instance of the receiving end system responses to the received data packets with acknowledgment packets indicating the buffer size available at its site and the sequence number of the last packet received from the sender. After receiving an acknowledgment packet for previously sent data the TCP sending instance deletes the local copy in the transmission buffer and can thus accept and send new data packets from the application.

The first version of TCP [143] addressed only the issues of reliability and flow control in the sense that it avoided sending more data than the receiver could handle. While this works fine for the case where the end systems are the bottlenecks, for the case where the network is the actual bottleneck this can have severe drawbacks. Nagle reported in [121] about a situation in the Internet where the network was highly utilized but with only a small amount of useful data reaching the receivers. This resulted from a severe congestion situation that lead to unnecessarily retransmitting data packets that were either in transit or had already arrived at the receivers. To account for the congestion situation in the network and aid the sender in choosing the appropriate transmission window size, the transmission window has now different adjustment phases [75]:

- **Slow start**: The slow start algorithm adds another window to the TCP source called the congestion window (CWND) which is measured in packets, i.e., in units of the size of the maximum packet size possible for this connection. When a new connection is established or after the expiration of a timer, this window is set to the size of one packet. Each time an acknowledgment is received, the CWND is increased by the size of the acknowledged packets. The source can now transmit up to the minimum of the CWND and the window size advertised in the acknowledgment packets.
- **Congestion avoidance**: Through the exponential increase of the sending rate caused by the slow start algorithm, the capacity of the network will be reached at some time and an intermediate router will start discarding packets. In order to avoid this, the slow start algorithm was supplemented through the congestion avoidance algorithm. With this algorithm the congestion window is increased for each acknowledgment as follows:

  $$\text{CWND} \mathrel{+}= \frac{1}{\text{CWND}}$$

  This way the congestion window increases by at most one packet each round trip instead of doubling it as was the case for slow start. For a more detailed description of the different congestion control mechanisms of TCP see [179].

Since the Internet collapse [121] various versions of TCP with congestion control enhancements were discussed. To mention only the most important versions:

Tahoe-TCP: First implemented in 1988 in 4.3 BSD Tahoe-TCP and was based on the work of Jacobson in [75]. Tahoe-TCP implements slow start, congestion avoidance and fast retransmission. In the Tahoe-TCP version, TCP reacts to a packet loss by setting a variable called the slow start threshold (ssthresh) to half of the current congestion window (CWND) and decreasing CWND to one. After receiving the acknowledgment for the retransmitted packet the source enters the slow start phase and the transmission window can be increased exponentially while (CWND <= ssthresh).

Reno-TCP: Implemented in 1990 in the 4.3 BSD Reno-TCP and is currently the most widely used version of TCP. In the Reno-TCP version, the fast recovery scheme is used in addition to the slow start, congestion avoidance and fast retransmission used in Tahoe-TCP. With the fast recovery mechanism the source retransmits the lost packet after receiving three duplicate acknowledgment packets and sets ssthresh to half of the current congestion window (CWND). CWND is then set to ssthresh plus 3 times the packet size. Each time another duplicate acknowledgment is received, CWND is incremented by the packet size and a packet is transmitted if allowed by the new value of CWND. When the next acknowledgment arrives that acknowledges new data CWND is set to ssthresh and the congestion avoidance mode is entered.

New-Reno: As Tahoe-TCP reduces the transmission window down to one after detecting a loss it is rather conservative and might lead to underutilization of the network. On the other hand, in the case of multiple losses per round trip time Reno-TCP results in poor performance as it might cause the reduction of the transmission window several times in a row or might even lead to halting the transmission until a timeout expires. Hoe et al. [68] present some enhancements to the fast recovery mechanism of Reno-TCP that allows the sender to better detect the loss of a row of packets and avoids reducing the congestion window several times in a row.

SACK-TCP: With the selective acknowledgment TCP (SACK-TCP) [109] the receiver includes in its acknowledgment packets a number of blocks each indicating a non-contiguous set of data that was received. This allows the sender to detect several losses at once and can avoid unnecessary delays and retransmissions and hence, achieve improved throughput.

Vegas-TCP: With Vegas-TCP Brakmo et al. [23] describe an improved approach for determining the window sizes using variations in the measured round trip delays.

ECN-TCP: With early congestion notification (ECN) network routers indicate overload situations in the network by setting a congestion bit in the packet headers. Floyd et al. [52] and Sisalem et al. [171, 170] propose various enhancements that allow TCP to react to overload situations before packets get dropped. While this approach reduces the loss probability in the network and improves the overall throughput it requires support from the network routers. Hence, while the other versions of TCP can be gradually introduced by upgrading the end systems only, ECN-TCP requires the support of the network routers as well.

Analytical Model of TCP Floyd et al. [56] and Mathis et al. [110] propose a model that estimates the throughput of a TCP connection (r_{TCP}) under known delay and loss conditions as

$$r_{\mathrm{TCP}} = \frac{C \times M}{t_{\mathrm{RTT}} \times \sqrt{l}} \tag{2.5}$$

with M as the maximum packet length, t_{RTT} as the round trip delay of the connection and l as the average loss measured during the lifetime of the connection. C is a constant that is set to either 1.22 or 1.36 depending on whether the receiver acknowledges one or more packets with each acknowledgement packet.

This TCP-throughput model is based on a TCP sender interpreting packet drops as an indication of congestion and responding by reducing the congestion window and thereby the effective sending rate by half. Further, the sender increases its transmission window by at most one packet each round trip time.

While this model is rather simple, it is only partially correct. It does not consider timeout cases or delayed acknowledgments. However, even under those restrictions different studies [56, 110] have shown this model to be accurate enough for losses of less than 5%.

For a TCP-model to be accurate over a wider range of losses it needs to consider timeout situations as well. Padhye et al. present in [129] an analytical model for determining the average transmission rate of a TCP sender (r_{TCP}) under known losses (l) and round trip delays (t_{RTT}).

$$r_{\mathrm{TCP}} = \frac{M}{t_{\mathrm{RTT}}\sqrt{\frac{2Dl}{3}} + t_{\mathrm{out}} \min\left(1, 3\sqrt{\frac{3Dl}{8}}\right) l \left(1 + 32l^2\right)} \tag{2.6}$$

with M as the packet size, t_{out} as the TCP retransmission timeout value and D as the number of acknowledged TCP packets by each acknowledgment packet.

The work done in [129] shows using simulations and measurements that Eqn. 2.6 estimates the average achievable TCP throughput over large intervals of several seconds under various conditions and heavy losses rather correctly.

2.1.2.2 The User Datagram Protocol (UDP)

While TCP is a connection-oriented protocol with an explicit setup and termination phase, the UDP protocol is connection-less and provides a much simpler service. It just sends packets of data called datagrams from one host to the other but with no guarantee that the sent datagrams will actually reach their destination. Such a service is called a *best effort* service.

As TCP requires acknowledging each sent packet, for the case of a point-to-multipoint communication sending data to a large number of receivers would result in an acknowledgment implosion at the sender. Hence, multicast is realized solely with UDP in the Internet.

Additionally, TCP senders might have to stop transmitting data for a while awaiting an acknowledgment packet or have to reduce their transmission rate drastically when observing the loss of a packet. This behavior might be acceptable for data transfer applications for which the overall transfer time is the crucial QoS parameter. For applications that consume and playout data at regular intervals such as audio and video application such a behavior is naturally unacceptable. Hence, recently proposed real-time multimedia services such as IP-telephony and video conferencing applications will be based on UDP. However, as UDP does not support reliable transport of data or congestion control these features need to be provided by the application layer.

2.1.3 The Application Layer and Application Related Control

The applications are responsible for generating data content such as video images, voice or text and initiating and controlling the communication. Data generated by the sending end system's application is usually only processed and used by the application at the receiving end system without any interaction of the network nodes, i.e., end-to-end communication. However, there are some proposals for networking devices that implement application-based control. That is, based on the data content generated by the application the network devices do certain actions like dropping the packets or treating them preferentially [202, 184]. Currently, most of the network nodes are mainly responsible for forwarding and routing data in the Internet and thus the lines connecting the applications to the network nodes in Fig. 2.1 describe more of a work in progress.

Additionally, applications might use an application protocol for realizing application specific tasks such as marking the start and end of application units, e.g. video frames and providing the end systems with information about the data content. An example for such an application level protocol is the real time transport protocol (RTP) [158] widely used in conjunction with multimedia applications in the Internet.

2.1.3.1 The Real-Time Transport Protocol (RTP)

The real time transport protocol (RTP) [158] designed within the Internet Engineering Task Force (IETF)[1] is the most widely used application layer protocol for real time communication in the Internet. Most of the used conferencing application such as VIC [112] or VAT support RTP. Additionally, the standards proposed for the Internet-based telephony such as H.323 [72] or SIP [63] define RTP as the application level transport protocol for the data.

RTP is an end-to-end protocol that is often used together with other transport protocols, in particular UDP. RTP has no notion of a connection; it may operate over either connection-oriented (say, ATM AAL5) or connection-less lower-layer protocols (typically, UDP/IP). It does not depend on particular address formats and only requires that framing and segmentation is taken care of by lower layers. RTP offers no reliability mechanisms.

[1]The Internet Engineering Task Force (IETF) is a large open international community of network designers, operators, vendors, and researchers concerned with the evolution of the Internet architecture and the smooth operation of the Internet

It is typically implemented as part of the application or as a library rather than integrated into the operating system kernel.

RTP sessions consist of two streams, namely a data stream for audio or video data packets and a stream of control packets (using the sub-protocol called RTCP). In general, data and control streams use separate ports, however, they may be packed into a single lower-layer stream as long as RTCP packets precede the data packets within the lower-layer frame. A single stream may be advantageous for systems where flows may be costly to manage, e.g., ATM PVCs.

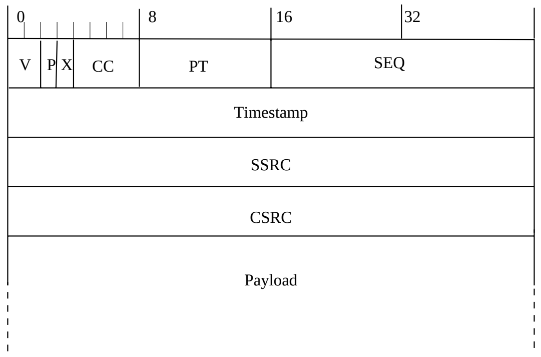


Figure 2.3: RTP data header

RTP adds to the data an additional header, see Fig. 2.3, that is sent as part of the data payload of the transport layer. RTP data headers include the following fields:

Version field (V): This field indicates which version of RTP is being used whereas version 2 is the common one.

Extension bit (X): The X bit whether an extension header follows the fixed header or not.

CSRC count field (CC): The CC field contains the number of CSRC identifiers following the header.

Payload type field (PT): This field is used to dynamically distinguish between different audio and video encodings.

Marker bit (M): The marker bit is used to delineate application level data units such as video frames and audio talk spurts.

Sequence number (SEQ): The sequence number is incremented by one for each RTP data packet and can be used by the receiver for detecting losses.

Timestamp: The timestamp reflects the sampling instance of the first octet in the RTP data packet.

SSRC, CSRC: These fields contain the identity of the sending source.

For monitoring the QoS of a session or tracing the number and identities of the members in a session each session member periodically sends RTCP control packets to all other session members using the same distribution mechanism used for the data packets.

Each RTCP packet begins with a fixed part similar to that of the RTP header followed by one or more parts carrying a variety of information such as:

RR: Receiver reports (RR) consist of several entries with each corresponding to one active sender. Each entry contains information about the percentage and number of the lost packets observed in the stream from the corresponding sender, the highest sequence number seen and timing information.

SR: Sender reports (SR) include information about the amount of sent data and when this report was generated. Additionally, when the sender is also a receiver it includes reports about the received data in an identical way to the receiver reports.

SDES: The source description packets (SDES) include the identification information about the session member.

Bye: This packet indicates the end of a member's participation.

APP: The application packets (APP) contain application specific information and can be used for experimental purposes.

RTCP is designed to allow applications to scale automatically over session sizes ranging from a few participants to thousands. For each session the data traffic is subject to an aggregate limit called the *session bandwidth* of which a certain percentage, usually around 5%, is to be divided among all participants. Based on the length of the RTCP packets and the number of members each participant can determine the interval between sending two RTCP packets. To avoid having bursts of RTCP packets the interval is set to a minimum of 5 seconds. The interval is further varied randomly over the range of [0.5, 1.5] times the calculated interval to avoid unintended synchronization of all participants.

The senders can estimate the round trip delay to the receivers using the RTCP messages. The senders include in their RTCP packets a timestamp indicating the time the report was generated. For each incoming stream the receivers send a report indicating the timestamp of the last received sender report (t_{LSR}) for that stream and the time elapsed in between receiving the last sender report and sending the receiver report (t_{DLASR}). Knowing the arrival time (t) of the RTCP packet the sender can calculate the round trip time (t_{RTT}) as follows:

$$t_{\mathrm{RTT}} = t - t_{\mathrm{DLSR}} - t_{\mathrm{LSR}} \tag{2.7}$$

This calculation requires no synchronization between the clocks of the sender and receiver and is therefore rather accurate.

2.1.3.2 Conceptual Model of Multimedia Applications

While web and file transfer are still the prevalent applications in the Internet today multimedia communication is ever more gaining in importance. This is especially alleviated by the wide availability of low-cost multimedia end systems and simple architecture of the Internet that allows for group communication. The first trials for using the Internet for audio conferencing were reported at the beginning of the nineties [30]. Since then, multimedia conferencing over the Internet has gained considerably in popularity. In addition to live conferences and the streaming of pre-recorded audio and video contents by numerous Internet radios [154] there is currently a wide interest in various standardization and commercial groups in merging a part of the current telecommunication structure with the Internet and providing Internet-based telephony [159].

The amount of bandwidth required for transporting the same multimedia content varies by using different compression styles or changing the quality of the compression. Taking the example of the H.261 [31] compression style, a picture is divided into blocks of 8×8 pixels. A discrete cosine transform (DCT) converts the blocks of pixels into blocks of frequency coefficients. These coefficients are quantized and encoded. With such a compression style adaptation can be realized by reducing the number of sent frames. That is, if a device is delivering 30 pictures/sec the sender would only compress and send a portion of those images. Another approach would be to change the accuracy with which the coefficients of the discrete cosine transformation are encoded, i.e., reduce the quantization factor. This would be equivalent to reducing the number of bits used to encode the pixels. By changing the frame rate or quantization factor, the sender can control its transmission rate at the cost of changing the perception quality of the transmitted content. Which approach to use depends highly on the content itself. So while changing the frame rate might be more appropriate for slow motion content like a video conference, changing the quantization factor might be more suitable for a sporting event. Bolot and Turletti discuss in [18] the effects of using either the frame rate or quantization factor on the achieved signal to noise ratio (SNR). Besides such quantitative measurements more qualitative aspects need to be considered in the rate adaptation process as well.

In addition to H.261 there is a wide variety of compression styles for both audio and video that can dynamically adjust the amount of bytes required for coding some content based on the available resources, see [193, 70, 114, 111, 190] for surveys and examples of such work.

Further, to satisfy the needs of heterogeneous receivers requesting data content at different rates, some compression styles can encode the multimedia content in different flows or layers. As an example for a layered coder, the MPEG-2 [65] standard offers four different so-called “scalability modes” [27]: spatial, SNR, and temporal scalability, as well as data partitioning. MPEG-2 allows the simultaneous use of up to two different scalability modes (except for data partitioning) in any combination, hence resulting in a 3-level representation of the signal. In spatial scalability, the base and enhancement layers operate at different spatial resolutions. In SNR scalability, adjacent layers have the same resolution and the enhancement refines the quantization process performed in the base layer. In temporal scalability, the enhancement layer increases the number of frames per second of the base layer. Data partitioning splits encoded data into two bit streams, one containing more and the other containing less important data. More important data are low frequency coefficients and motion vectors.

In this work, we will primarily be addressing the issue of determining the bandwidth share a multimedia flow should use. How the actual determined behavior is realized at the end systems, e. g., the decision of adjusting the frame rate, quantization factor of a video stream or the shape of the layers to use will be beyond the scope of this work. That is, we will be assuming that the sender is capable of adjusting the compression parameters in a way as to restrict the produced data stream to the calculated bandwidth share this sender can utilize.

Usually, there are certain maximum and minimum bandwidth requirements for multimedia contents above and below which the transmitted data is of no use for the receiver. Additionally, the used compression style for a multimedia content might dictate that the changes in the transmission rate of a video stream for example can only be realized in a granular way. These aspects depend on the content and compression style used. Unless explicitly mentioned, in the most part of the work we will ignore these restrictions and assume that the end systems are able to adapt their transmission and reception behavior exactly to the level determined by the adaptation algorithm. The effects of considering user, content or compression imposed restrictions on the adaptation process will be investigated separately in chapter 5.

2.1.4 The Host-Network Interface

This part normally includes the device driver in the operating system and the corresponding network interface in the end system. Together they handle all the details of interfacing with the physical layer. Note, that the Internet is not based on any specific hardware or link-layer protocols.

2.2 End-To-End Estimation of Path Characteristics in the Internet

In this study, we will be discussing the issue of adjusting the sending and reception behavior of end systems in the Internet in accordance with the available resources and load situation in the path connecting those end systems. Typically, the routers in the network are the only components aware of availability of resources and the exact load situation. However, in the current Internet architecture the routers do not provide the end systems with such information. Hence, the end systems need to estimate the characteristics of the network such as the loss, delay and capacity using methods that do not actively involve the network routers. Based on this information, the end systems can then adjust their behavior in accordance with the estimated network load situation.

2.2.1 Estimation of Round Trip Time

Round trip time (T_{RTT}) indicates the time a data packet requires to go from one end system to the other and back. This time includes the propagation delay over the physical link and the time spent in the buffers of the routers as well as the transmission and processing time at the end systems.

A straight forward approach for estimating the round trip delay between two points (A and B) in the Internet is to send a data packet from point A to point B with the packet indicating the time the packet was transmitted (T^A_{send}) as measured by A. After receiving this measurement packet, B sends the received packet back to A which receives it at time (T^A_{rec}). The round trip time (T_{RTT}) can then be estimated as

$$T_{\text{RTT}} = T^A_{\text{rec}} - T^A_{\text{send}} \tag{2.8}$$

Measuring the one way delay (T_{OW}) between two end systems is, however, more complicated. Due to the possible asymmetry of the links connecting two end systems in both directions the one way delay can not be simply estimated as half the round trip delay. To estimate the one way delay, the sender (A) can send a packet with a timestamp T^A_{send} to receiver (B) that receives it at time T^B_{rec}. The one way delay can then be estimated as

$$T_{\text{OW}} = T^B_{\text{rec}} - T^A_{\text{send}} \tag{2.9}$$

However, clocks of distributed end systems are usually not synchronized and run at different speeds. In this case, the values of T_{OW} estimated using Eqn. 2.9 is useless and could actually be negative. Distributed clocks can be synchronized using either hardware or software systems. For highest possible accuracy, end systems need to be enhanced with appropriate hardware for realizing the client part of the global positioning system (GPS) [134]. With GPS, the client part contacts a satellite-based radio navigation system that provides three dimensional positioning, velocity and time information.

Due to the costs and complexity introduced by additional hardware systems distributed clocks in the Internet are often synchronized using a software based approach called the network time protocol (NTP) [115]. With NTP, a software process at the end system contacts an NTP server to get an estimation of the correct time, called universal coordinated time (UTC[2]). To reduce the load on the servers there are several NTP servers in the Internet. These servers communicate with each other to estimate the correct UTC value.

In addition to the synchronization problem, clocks of different systems run at different speeds which causes the clocks to drift apart, that is have a clock skew [118]. Thus even after synchronizing two clocks, the clock skew causes inaccuracies in the measurements.

2.2.2 Estimation of Losses

Due to an overload situation in the network some intermediate routers might start discarding packets instead of forwarding them. This can be detected by introducing sequence numbers in the data packets. Noticing a gap in the sequence numbers of the received packets at the receiver, a loss event can be assumed. The occurrence of such an event can be directly forwarded to the sender by either sending a negative acknowledgment to the sender or withholding a positive acknowledgment as is the case with TCP. That is, with TCP [179], after a loss the receiver keeps on sending acknowledgments of the last correctly received packet for each newly incoming packet.

[2]This is a time scale based on atomic clocks, by definition cesium clocks.

For the case that the sender is more interested in a summary of the losses, the receiver calculates the number of loss events N_{loss} during some time interval. Comparing this value to the actual number of received packets the receiver gets an estimation of the network losses. This value can then be conveyed to the sender using a control packet as is the case with RTP [158]. In this case some care needs to be taken in setting the length of the time interval over which losses are measured. Choosing the measurement interval too small might result in highly oscillative values. On the other hand, large intervals might lead to underestimations of the loss and neglect the possibly bursty nature of the losses.

2.2.3 Estimation of Bottleneck Bandwidth

The bottleneck bandwidth of a path between two end systems indicates the bandwidth of the network router with the smallest capacity along this path. The bottleneck bandwidth of a path determines, hence, the maximum bandwidth share a flow traversing this path can assume.

The bottleneck value can be obtained using the packet-pair approach first used by Bolot [15] for estimating the characteristics of Internet paths. The essential idea behind the packet-pair approach is [75]: If two packets can be caused to travel together such that they are queued as a pair at the bottleneck, with no packets intervening between them, then the inter-packet spacing will be proportional to the time required for the bottleneck router to process the second packet of the pair.

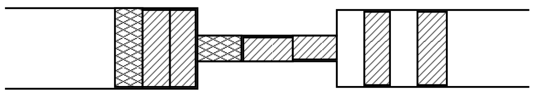


Figure 2.4: Illustration of packet flow through a bottleneck

Fig. 2.4 illustrates how two packets arriving back to back at a link get spaced out. The resulting time gap between the two packets is then preserved even over links with a higher capacity. By sending two packets of size (P) each at the access speed of a station, the packets get spaced out at the bottleneck router and arrive at the receiver with time gaps of G seconds between the two packets. Thereby the bottleneck bandwidth (B) can be estimated as

$$B = \frac{P}{G} \tag{2.10}$$

Carter et al. [29] present a tool called BPROBE for measuring the bottleneck bandwidth using this probing approach. With BPROBE a sender transmits a stream of ICMP ECHO [38] packets to some target. These packets get echoed at the target and sent back to the sender. The source can then estimate the bottleneck bandwidth by measuring the interarrival times of the returning packets. In a similar approach, Jacobson implemented an application called PATHCHAR[3] [43] for estimating the characteristics of Internet links

[3]The tool is freely available from ftp://ftp.ee.lbl.gov/pathchar/

such as the bottleneck bandwidth and the round trip delays. PATHCHAR takes advantage of the time to live field (TTL) in an IP packet. The TTL determines how many links a packet can traverse before it expires. Each router traversed by the packet, reduces the TTL value in the IP header by one. If a router receives a packet that has expired it drops the packet and sends an ICMP error packet back to the sender. The source address of the router indicates which router dropped the packet. Again by measuring the interarrival times of the ICMP error packets the sender can estimate the bottleneck bandwidth of the path towards the router which dropped the packet.

Different factors can effect the accuracy of the probing scheme:

Queuing failure: Packets might not be queued at the router which results in overestimating the bottleneck bandwidth. BPROBE and PATHCHAR solve this problem by using a row of probing packets and packets of large sizes which increases the possibility of the packets getting queued.

Competing traffic: Packets of some other flows might interfere with the row of probe packets. This would then lead to larger gaps between the probe packets and hence underestimating the bottleneck bandwidth. To avoid the negative effects of traffic interference BPROBE and PATHCHAR use long streams of probe packets to increase the possibility that two probe packets do get queued in a row. Additionally, filtering mechanisms are required to identify and neglect wrong measurements.

Network congestion: If after passing the bottleneck router the probe packets get queued at another router, the gab between the probe packets will be reset and describe the capacity of the wrong link. This effect needs to be taken care of with the filtering mechanisms.

With BPROBE and PATHCHAR the sender estimates the bottleneck bandwidth based on the interarrival times of the echoed probe packets. As these echoed packets might suffer the same problems of queuing failures, interfering traffic and network congestion the number of wrong measurements is further increased. Paxon [137] presents therefore another approach in which the receiver of the probe packets determines the bottleneck bandwidth and informs the receiver of this value. With this approach Paxon [137] achieves more accurate results than with the echoing approach. Lai et al. [97] further propose refined statistical approaches for improving the accuracy of the filtering mechanisms and reducing the possibility of wrong estimations.

2.3 Investigation Environment

The investigations of the algorithms and mechanisms discussed in this work will largely be based on simulating the algorithms under different network topologies and conditions. Where possible, the simulations will be verified by actual implementation and tested over the Internet.

2.3.1 Simulation Models and Environment

The simulations were realized using the PTOLEMY [99] simulation tool[4]. PTOLEMY provides the user with a graphical interface for creating and controlling discrete event simulations [79]. The user can further add additional modules.

For investigating RTP-based adaptation mechanisms and their effects on TCP we extended PTOLEMY with additional modules for TCP and RTP as well as various modules for simulating the Internet with its routers and traffic sources.

The RTP Simulation Model Multimedia applications using RTP and UDP are simulated using an RTP model. This model simulates the behavior of RTP senders and receivers that can transmit data packets at a speed determined by the used adaptation algorithm. Each packet has the basic RTP header including timing information and the identity of the sender. Additionally, the end systems exchange RTCP reports as was described in Sec. 2.1.3.1.

The model provides the user with an interface for specifying the transmission rate, the end system's identity and the session bandwidth to use. Detailed information can be obtained from [46].

The TCP Simulation Model For simulating the behavior of TCP we implemented two versions of TCP: Tahoe-TCP with fast retransmission, delayed acknowledgments and round trip calculation and the Reno-TCP version with the fast recovery scheme as well. Both versions use maximum transmission windows larger than the 64 kbytes allowed with the current implementations of TCP [76]. As the round trip time values in some of our simulations are in the order of just a few milliseconds, the coarse-grain timer used in Unix TCP (typically, a granularity of 500 msec is used) would result in imprecise timeout values. Therefore, we used here a clock granularity of 200 msec. As the Reno-TCP model is the currently most widely used version in the Internet most of our simulations using TCP in the work will be based on the Reno-TCP model.

While the simulation model is not based on production code we have run various simulations for verifying the model's correctness and compared the results to available results in the literature [161]. Detailed information can be obtained from [163].

The Internet Simulation Model Simulating how the global Internet data network behaves is an immensely challenging undertaking because of the network's great heterogeneity and rapid change. The heterogeneity ranges from the individual links that carry the network traffic, to the protocols that interoperate over the links to the different applications used over the network [139].

In this study, we are, however, only interested in partial aspects of the Internet, that is of the interaction of TCP-based applications and adaptive multimedia application as well as the capacity of the network in terms of link bandwidth and available buffers in the network nodes. Multimedia applications can be simply modeled as RTP streams with the

[4]see: http://ptolemy.eecs.berkeley.edu/

senders continuously transmitting data. TCP applications can be divided into different categories:

Active web traffic: The world wide web (WWW) is the most widely used TCP application. The share of web traffic in the Internet has seen a tremendous increase from practically non-existent a few years ago up to around 70% of the number of connections and packets in the Internet today [35]. Various studies investigating Internet traffic in general [136, 137] and web traffic in particular [133] reveal that web traffic can not be simply modeled using Poisson models [138] but is rather self-similar. Self-similarity is manifested in the absence of a natural length of a burst; at every time scale ranging from a few milliseconds to minutes and hours, bursts consist of bursty sub-periods separated by less bursty sub-periods [58]. Self-similarity can be constructed based on aggregating many simple renewal reward processes exhibiting inter-renewal times with infinite variances (heavy tailed distribution). For the case of web traffic such an effect can be generated by aggregating the traffic from a number of web servers with each server starting a TCP connection for transferring a number of packets, going afterwards into an idle period and then starting gain. The number of packets to be transferred is drawn from a heavy-tailed distribution with

$$P[X > x] \sim x^{\alpha} \text{ as } x \to \infty$$

where $0 < \alpha < 2$. One simple heavy-tailed distribution is the Pareto distribution. The Pareto distribution is power-low over its entire range; its probability density function is given by

$$p(x) = \alpha k^{\alpha} x^{-\alpha-1}$$

where $\alpha, k > 0$ and $x \geq k$. Its distribution function has the form

$$F(x) = P[X \leq x] = 1 - \left(\frac{k}{x}\right)^{\alpha}$$

With $\alpha \leq 2$ the Pareto distribution has infinite variance.

Measurements done on web traffic [133, 35] suggest that the length of a web connection can be drawn from a Pareto distribution with a scale factor (α) of 1.1 and a mean of 20 packets. Active transfer phases are then followed by idle periods drawn from a Pareto distribution with a scale factor of 1.8 which results in a behavior more similar to the exponential distribution and a mean of 0.5 seconds.

Background web traffic: Similar to the active web traffic, the background web traffic is modeled as the aggregate of a large number of on-off processes with the on period lasting for the time needed to carry a number of packets drawn from a Pareto distribution with the factor of 1.1 and a mean of 20 packets and the off period lasting for a time drawn from a Pareto distribution with a factor of 1.8 and a mean of 0.5 seconds [133].

With the active web traffic each burst consists of a TCP connections that lasts for the duration needed for transporting the packets of the data burst. Hence, the aggregated traffic bandwidth share depends on the congestion situation in the network. With the background traffic, the packets of each burst are sent at a constant rate specified by the user. This kind of traffic is useful for populating the network with bursty and uncorrelated traffic with a defined maximum rate.

FTP connections: The file transfer protocol(FTP) [144] is a popular application for transferring large data files between two sites. While the share of FTP traffic in the overall TCP traffic has decreased considerably since the introduction of the world wide web it still occupies around 5% of the overall Internet traffic. While web traffic is characterized through small bursty transfers, FTP connections tend to last for longer time periods and carry a much larger number of packets.

In addition to the traffic sources we, further, modeled various necessary components such as routers with RED and FIFO buffer management mechanisms and a basic multicast protocol for joining and leaving a multicast session as well as establishing a shortest path multicast routing tree.

As a general test case we use a dumb-bell topology, see of Fig. 2.5, consisting of a bottlenecked link connecting m RTP-based senders and receivers, n TCP connections carrying FTP traffic and k TCP connections carrying active WWW traffic. The link has a capacity of R Mb/s and a round trip propagation delay of τ. The end systems are connected to the bottlenecked router with a link of capacity R Mb/s and a propagation delay of τ_{conn}. For the rest of this study τ_{conn} will be arbitrarily set to the negligible value of 1 μsec. The bottlenecked router (Router$_1$) can introduce a maximal additional delay of τ_{q} due to the buffering of the data. The second router (Router$_2$) servers only as a distributer of the data to the end systems and does not introduce losses or delays to the transported data.

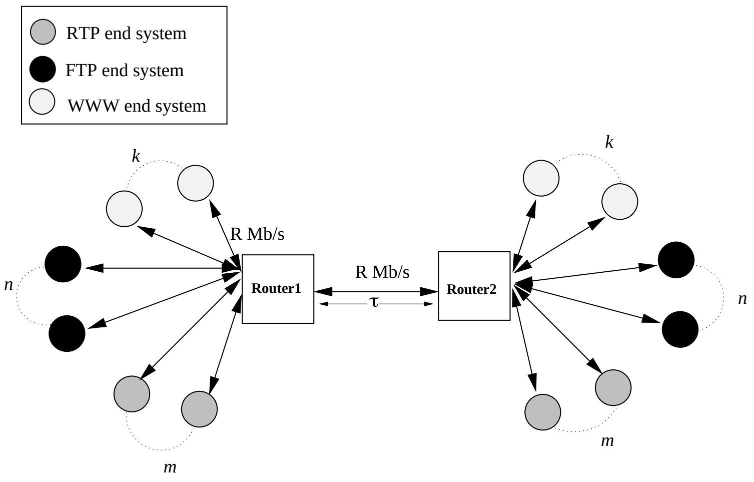


Figure 2.5: Adaptation performance testing topology

Each simulation is run until the sum of the bandwidth consumed by each traffic type exhibits steady state behavior. To account for the effects of the transient period, the first 200 seconds are ignored in the performance calculation. Each simulation was run four times and the presented values are the average of the values reached in the different runs. The FTP as well as the RTP senders are greedy senders with an infinite amount of data to send. The WWW connections are modeled as active web traffic

As the buffer management approach in the simulations we use both the random early drop scheme (RED) and FIFO, see Sec. 2.1.1.1. For all simulations using RED the minimum buffer threshold is set to 30% of the available buffer and the maximum threshold to 80%. The queuing weight (w_{q}) is set to 0.002 and the maximum drop probability (P_{a}) to 0.02. These values were chosen based on the guidelines presented in [55]. Finally, unless otherwise specified, the packet size is set to 1000 bytes.

In addition to this topology, a number of other topologies used for testing more specific cases such as multicast will be introduced separately when needed.

Chapter 3

Traffic Management in the Internet

The flexible architecture of the Internet and the decline in bandwidth prices have made it possible to use the Internet for various multimedia applications such as telephony and conferencing.

However, transmitting data packets into the network without regard to the actual available resources can easily result in overload situations and data losses. To avoid such a situation some mechanisms are required to control the amount of data packets entering the network.

In the literature there have been generally two approaches for traffic management in a network: admission control combined with QoS signaling protocols on the one hand and congestion control schemes that adapt the transmission and reception behavior of the end systems in accordance with the available resources in the network on the other.

In Sec. 3.1 a general survey of congestion control mechanisms for unicast and multicast communication is given as well as some evaluation criteria of congestion control schemes. Sec. 3.2 describes briefly different proposals for quality of service (QoS) architectures that provide for tighter control on the number of admitted flows to the network and the amount of network resources received by each flow. In Sec. 3.3 a short overview of the advantages and disadvantages of congestion control schemes compared to QoS architectures is given.

3.1 Congestion Control in the Internet

As the Internet does not provide for any mechanisms for regulating the transmission behavior of sending end systems, a single misbehaving end system sending data at a higher rate than the available bandwidth might lead to severe unfairness problems at best case and to network congestion collapse at the worst [53].

The unfairness problems arise from the existing architecture of the network elements in the Internet. As already mentioned in Sec. 2.1.1.1, currently, most of the network nodes, commonly known as routers, in the Internet deploy tail discard queuing systems [98], mainly due to their simplicity. With such an approach an end system sending a data stream at a high rate competing with an end system sending at a lower rate will have the same loss probability as the low rate sender. Hence, the high rate stream actually receives a larger share of the available bandwidth than the low rate stream even though it actually caused the congestion [53]. This scenario is even worse for the situation of TCP connections sharing the same bottleneck with UDP flows. In response to the loss of packets, TCP connections reduce their transmission rate, see Sec. 2.1.2. However, without any rate reduction on behalf of the non-congestion-controlled UDP traffic sharing the same bottleneck with the TCP traffic, the congestion situation would prevail and the TCP connections would starve and receive a much smaller bandwidth share than the non-responsive flows [53].

Severe and consistent congestion states might also lead to congestion collapse. Such a situation occurs when the network is utilized to a very high degree but with only a small amount of useful data reaching the receivers. Congestion collapse in the Internet was first reported by [121] and was largely due to TCP connections unnecessarily retransmitting data packets that were either in transit or have already arrived at the receivers. This form of collapse has since been avoided by different improvements in the TCP congestion avoidance mechanisms [75].

Other forms of congestion collapse arise when the network transmits fragments of packets that will be discarded at the receiving end systems because they can not be reassembled into a valid packet. Such a situation might occur from the mismatch between the link level transmission layer and higher transmission layers. For example when carrying large IP packets over an ATM network, dropping an ATM cell might lead to the discard of the entire IP packet at the receiver. Such a situation might be avoided by mechanisms aimed at providing lower layers with knowledge about the fragmentation of data in higher layers. The early packet discard [149] (EPD) is such an approach. Another approach is Path MTU Discovery [94] which aims at minimizing fragmentation.

The congestion collapse form that we are interested in, in this study, arises when network resources are wasted by transporting packets through the network that are dropped before reaching the receiving end system. This situation arises when the sender is transmitting data at a higher rate than the rate of some bottleneck on the path towards the receiver. Such a situation can be avoided by adjusting the transmission rate of the sender in accordance with the available network resources.

The issue of congestion control in data networks has received great attention in the literature [201]. Citing only a fraction of the proposed work is, however, far too voluminous to fit in this work. Therefore, we first give a brief overview of classification possibilities of congestion control schemes and only present some examples for those classes which will be studied later in this work.

Depending on their design goal, architecture, applications and the environments they were designed for and various other issues, congestion control schemes can be classified in different ways. To mention only a few classification possibilities:

Control loop: Depending on whether or not the sender needs to collect information about the network path connecting it to the receiver in order to adjust its transmission behavior, we can distinguish two control possibilities:

Open-loop adaptation: The sender adjusts the shape of the data to be transmitted in accordance to prior knowledge or estimation of the available resources. For example, when transmitting live video with variable bit rate over a constant rate link, the sender can adjust the quality or frame rate of the video stream to be suitable for transmission over the available network capacity. This is often realized by using a leaky bucket [100] for enforcing a specific transmission rate. By monitoring the buffer length of the leaky bucket, the application can determine the appropriate quality factor to send the data with [106]. Additionally, instead of altering the transmission behavior the sender might in a different scenario adapt the priority of its data packets in accordance to their content and the available resources [49]. Sanneck presents in [151] an approach for improving the quality of audio communication over the Internet. In addition to the audio data the sender includes information that can be used for the concealment of lost packets. The amount of additional information needed for reconstructing the original content is adapted to the importance of the sent audio data.

Closed-loop adaptation: With open-loop adaptation the sender can only adjust its behavior based on prior knowledge of its own resources or of the static amount of network resources available to it. Here, the aim is to maintain or improve the quality of the communication in accordance with an amount of fixed resources. To actually avoid network congestion the sender needs information about the path connecting it to the receiver and back. Such a closed-loop adaptation is a more general case where the sender adjusts its transmission behavior not only based on the locally available resources but based on feedback information from the receivers [25] or the network [162]. This approach has the major advantage of avoiding network congestion or overwhelming the receiving end-system with data it can not use. Depending on the communication scenario there is a multitude of possibilities for realizing closed-loop adaptation.

Transmission mode: Depending on the number of participants in a communication scenario different issues need to be considered when designing congestion control schemes.

Unicast: For the case of point-to-point communication tight control can be realized. Similar to the case of TCP, the receiver of a multimedia flow can send frequent feedback messages to the sender without overloading the sender. This would then allow for fast reactions to changes in the network state.

Multicast: Having a large number of receivers sending frequent feedback messages to the sender can easily result in overloading the sender or might cause congestion on the link connecting the sender to the network. Hence, for the case of multicast, scalability issues play a crucial role in the design of congestion control schemes. Additionally, due to the heterogeneity of the Internet links and receivers congestion control schemes need to be carefully designed so as to achieve a high performance in a variety of situations.

Architecture: Depending on the systems involved in the congestion control scheme different modes can be distinguished:

End-to-end: In this case only the end systems, i.e., the senders and receivers, are involved in collecting information about the congestion state in the network and adapting the number of packets entering the network.

Network-supported: In addition to the end systems, network routers could participate in collecting state information as well as manipulating the amount of data transported in the network.

Note, that these classes should only be considered as conceptual ones, that aid us in discussing different features of congestion control schemes. Actually, congestion control schemes usually belong to overlapping classes. Network-supported schemes can for example be either open-loop or closed-loop and be used for unicast as well as multicast communication. In this study, we will restrict the scope of investigated congestion control schemes to closed-loop schemes for unicast and multicast communication. While we will be mainly concentrating on the end-to-end communication scenario we will also be briefly looking at network-supported approaches. In Sec. 3.1.1, we first give some criteria for evaluating the efficiency of congestion control schemes. In Sec. 3.1.2 different approaches and schemes for realizing congestion control schemes for unicast are described. Various examples of congestion control schemes for multicast communication are then presented in Sec. 3.1.3. Sec. 3.1.4 presents a short survey of network-assisted congestion control schemes.

3.1.1 Evaluation Criteria for Congestion Control Mechanisms

By deploying congestion control mechanisms in the context of multimedia communication we aim at improving the network performance compared to the case when no control is deployed and thereby increasing the user's satisfaction. In this section, we will describe some of the evaluation criteria of the efficiency of the control mechanisms we will be investigating in later chapters.

3.1.1.1 Network Congestion State

The primary goal of congestion control schemes is to achieve a high network utilization level and at the same time reduce the packet losses at the network level. In estimating the loss performance of a scheme, one should consider the losses as seen at the networking layer and not only at the application layer. Using forward error correction (FEC) mechanisms [28] for reconstructing lost data packets might lead to loss free data transmission as measured by the application. However, in this case, network resources are wasted transporting redundant data. Hence, while FEC mechanisms improve the quality of single streams, using congestion avoidance mechanisms one aims at improving the overall performance of the network and thus all streams traversing the network.

3.1.1.2 Scalability

Based on the multicast architecture of the Internet a sender can transmit a media stream to a nearly unlimited number of receivers in an effective way. The performance of an efficient congestion control scheme should therefore not degrade substantially with the increased number of participants in a multimedia session. Additionally, the complexity of the schemes used should still be acceptable.

3.1.1.3 Convergence

Convergence is generally measured by the speed or time with which a system approaches some stable state from any starting state. Efficient adaptation algorithms need to have a short convergence period and lead to a stable state.

3.1.1.4 Fairness in the Context of Congestion Control for Multimedia Communication

When confronted with the case of connections or users competing for some shared resources such as network bandwidth we need to consider the aspect of fair resource distribution. However, the term fairness has only a subjective meaning that differs depending on the situation. Thus, a system is to be considered fair if the competing instances receive a share that conforms to some pre-established fairness definition.

In the context of multimedia communication there have been two major definitions of the values of a fair share a connection can receive: max-min fairness and TCP-friendly bandwidth distribution. Which definition to use depends generally on the control scenario and the information available for the adaptation process.

The Max-Min fairness

The max-min fairness criteria [77] is usually applied when the adaptation involves the distribution of available resources for a known number of instances, e.g. bandwidth for a number of streams or processing power for a number of processes.

Taking the example of bandwidth distribution among competing streams, whereas a stream indicates here the flow of data packets from one sender to one or more receivers, the fair share (F_i) of a flow i is minimally

$$F_i = \frac{B}{n} \tag{3.1}$$

with B as the bandwidth of a network node and n as the number of flows traversing this node. (F_i) can maximally reach

$$F_i = \frac{B - \sum B_{\text{cong}}}{n - \sum n_{\text{cong}}}. \tag{3.2}$$

with B_{cong} as the bandwidth of the flows bottlenecked elsewhere and n_{cong} as the number of flows bottlenecked elsewhere. A bottlenecked flow indicates that the fair share a flow can get at some other link is smaller than its minimum fair share at this link. A flow's fair bandwidth share is then set to the minimum fair share calculated at all traversed hops.

As an example, consider the topology depicted in Fig. 3.1. The first router has a capacity of 10 Mb/s and is shared between 4 connections, hence, the router would calculate

F_i to be 2.5 Mb/s. However, the second router traversed by the flows C_1 and C_2 has only a capacity of 2 Mb/s. Thus the fair share of the flows C_1 and C_2 is only 1 Mb/s. Now, if the flows C_3 and C_4 restricted their transmission rate to the fair share value, i.e., 2.5 Mb/s, 3 Mb/s of the bandwidth available at the first router will be wasted. Using the max-min fairness criterion the fair share of C_3 and C_4 should actually be 4 Mb/s.

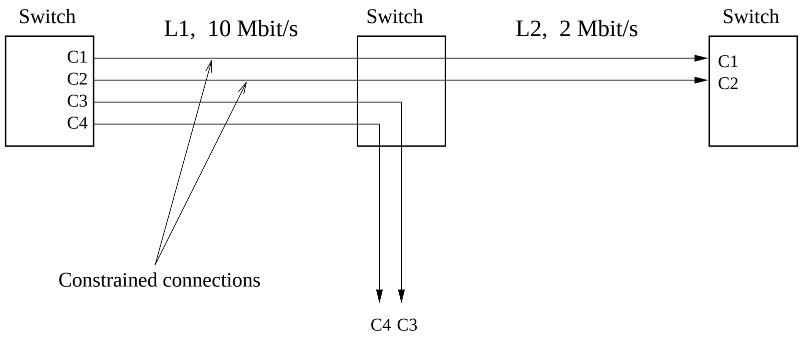


Figure 3.1: An example for max-min distribution

This definition can be further extended as follows:

Reservation proportional shares: With each user reserving -or paying for- a guaranteed share (S_i) of some resource R, each user (i) gets a fair share proportional to its S_i. For n flows the fair share F_i can be calculated as follows:

$$F_i = R * \frac{S_i}{\sum_i^n S_i}$$

Reservation plus equal shares: With each user reserving -or paying for- a guaranteed share S_i of some resource R, each user (i) gets its reserved S_i plus an equal share of the available resources. For n flows the fair share F_i can be calculated as follows:

$$F_i = S_i + \frac{R - \sum_i^n S_i}{n}$$

Weighted shares: Here, the resource (R) is allocated for each connection proportionally to a pre-determined weight W_i -e.g., priority level. For n flows the fair share F_i can be calculated as follows:

$$F_i = \frac{R * W_i}{\sum_i^n W_i}$$

Note, however, that applying these enhanced definitions requires a mechanism for establishing reservations or setting the allocation weights. As such mechanisms are currently not available in the Internet these definitions are inappropriate for control mechanisms realized only at the end systems. Thus, we will not consider these definitions further in this work.

In its specification of the available bit rate (ABR) service [162] the ATM-Forum[1] adopted the max-min fairness definition [77] as the basic fairness measurement.

For a quantitative description of fairness Jain [80] suggests using the so called fairness index:

$$F = \frac{\left(\sum_i x_i\right)^2}{n \sum_i x_i^2} \tag{3.3}$$

with x_i as the ratio of the actual share to the fair share that a flow should theoretically assume and n as the number of flows.

TCP-Friendly Adaptation

Often, it is the case that the adaptation process is based on an estimation of the performance of a system and the adaptation instance heuristically adjusts its behavior in accordance with the estimated performance without exact knowledge of the available resources or number of other competing instances. Adapting the transmission behavior of a video stream based on the measured loss and delay values in the network presents such an example. In this case, applying the max-min fairness criteria is rather difficult. Instead, we need to consider the effects of using an adaptation scheme on the shares received by other adaptive traffic. As around 95% of the traffic carried over the Internet is TCP traffic [35, 185] there has been a current consensus in various works [56, 57] that adaptation mechanisms and congestion control approaches should be *TCP-friendly*. TCP-friendliness indicates here, that if a TCP and an adaptive flow with similar transmission behavior traverse the same path and thus compete at the same bottleneck, have similar round trip delays and face the same loss values both flows should receive similar bandwidth shares. Due to the different natures of TCP and multimedia traffic it is expected that a TCP-friendly multimedia flow would acquire the same bandwidth share as a TCP connection only averaged over time intervals of several seconds or even only over the entire life time of the flow and not at every time point.

The requirements for fulfilling both the max-min fairness criteria and TCP-friendliness are, however, rather contradictory. With the max-min fairness definition the resource distribution depends only on the number of flows sharing a bottleneck. Various simulative [51, 171] as well as analytical [56, 128] studies have shown that the bandwidth a TCP connection can utilize depends not only on the number of connections traversing a shared link but also on the round trip delay and the number of traversed congested hops. That is, connections having large round trip delays or traversing a larger number of congested hops receive a smaller bandwidth share than a connection with a smaller round trip delay or less congested hops in its path. While the question whether connections with different round trip time delays and numbers of traversed hops and thus consuming a different amount of network resources should be treated equally is rather a philosophical one, there has been a common consensus [51] that this behavior is unfair. However, due

[1]The ATM Forum is an international consortium whose goal is to accelerate the use of ATM products and services through the development of interoperability specifications and the promotion of industry cooperations.

to the significant share of the TCP traffic in the Internet, we will primarily consider the aspect of TCP-friendliness when evaluating the fairness of adaptation schemes.

For a quantitative description of the TCP-friendliness of some scheme we use the following equation:

$$F = \frac{r_{\mathrm{adapt}}}{r_{\mathrm{TCP}}} \tag{3.4}$$

with r_{adapt} as the average rate achieved by a flow controlled by some adaptation scheme and r_{TCP} as the average rate of a competing TCP connection. Depending on the adaptation scenario these values can be measured over intervals as short as a few round trip delays or as large as the entire life time of the flow. This is the same measure used in other studies such as [130, 146].

3.1.1.5 User Satisfaction

Users' satisfaction is in general difficult to measure and quantify. There have already been some proposals for assessing the QoS of multimedia contents using rating schemes as was proposed in [197] and by the ITU [73]. Other studies propose automatic assessment approaches [188, 198].

However, in evaluating the performance of an adaptation scheme, users' satisfaction is just one evaluation criteria out of many. Aspects of network performance or scalability of the scheme are just as important. Due to the complexity of the measurement mechanisms we refer to a simple measurement of the loss ratio of the data presented to the user. While this might be an oversimplification of the evaluation procedure, we need to note that the mechanisms to be developed in this work are to be compared mainly to the situation of a multimedia communication without any adaptation involved. Studies on the transmission of MPEG video streams over the Internet have shown that losses are the major source of QoS degradation and that losses as low as 3% might render about 30% of a video stream unusable [20]. Hence an effective adaptation scheme should aim at minimizing the loss ratio observed at the users' end systems.

3.1.2 End-to-End Unicast Congestion Control Schemes

Basically, congestion control schemes for unicast communication can be divided in two major categories of algorithms, window-based and rate-based mechanisms. In rate-based adaptation, a limit is placed on the rate at which the source can send packets. In window-based flow control, at any time there is a limit to the number of packets a sender can transmit before having to wait for an acknowledgment from the receiver, but there is no constraints on the rate at which packets can be sent.

The adaptation decision itself can be based on either explicit or implicit indication about the available resources. Explicit information can be as detailed as the actual bandwidth share a sender should be using to avoid network congestion or simply a single bit indicating the congestion state of the network. Explicit indications are created by the network and either forwarded downstream on the flow's path towards the destination or sent directly back to the senders. With the forward error congestion notification (FECN), the sender transmits special messages that collect information from all traversed routers along the path to the receiver which echoes them back to the sender. This approach provides the destination with a complete overall look of the congestion state along the flow's path introduces, however, additional delay as the information needs to traverse all the path up to the destination before being sent back to the sender. FECN also minimizes the needed number of congestion messages compared to the backward error congestion notification (BECN). With BECN each intermediate router creates and send a congestion notification to the source end system when congestion is observed. This can be achieved by creating special messages to be sent to the end systems or by including the congestion notification in the messages that are sent from the receiver to the sender.

On the contrary, the network congestion state might also be implicitly estimated using loss and delay indications measured at the receiving end systems and carried in feedback messages sent by the receivers. In addition to losses and round trip delays the end systems might use the delay variations of the received packets as an indication of the buffer lengths in the network and hence as an estimation of the network load.

With the lack of mechanisms for collecting and distributing explicit information about the network congestion state in the Internet most of the control schemes found in the literature refer to heuristic approaches for adjusting the transmission rate or window, see Sec. 3.1.2.1 and 3.1.2.2. This heuristic approach is usually based on some variation of an increase/decrease mechanism (ID). With ID schemes, the bandwidth share of a flow is gradually increased by some amount during underload situations and reduced during overload situations. A popular version of ID schemes is the additive increase and multiple decrease mechanism (AIMD). With this approach the sender can increase its resource share by an additive amount during underload periods and reduces it by a multiplicative factor during overload periods. The choice of additive increase and multiplicative decrease can be briefly justified as follows. If the network is operating below the optimal point designated by high utilization and low losses, all users go up equally. If the network is congested, the multiplicative decrease makes users with higher resource shares go down faster than those with smaller shares making the allocation more fair. In [34] Chiu and Jain show that this approach leads to a stable adaptation behavior and an equal bandwidth distribution. As in the environment we are considering here the network does not provide for explicit notification about the available resources, our work will be mainly based on the increase/decrease concept using information about the losses and delay values measured at the end systems. In Chapter 7, we will be considering the benefits of using explicit information about the available bandwidth share.

3.1.2.1 Window-Based Adaptation

There has been several proposals for adaptation schemes that, similar to the TCP adaptation mechanisms, are based on the sliding window approach, see Sec. 2.1.2.

With the sliding window approach a window represents the number of packets the sender can transmit before having to wait for an acknowledgment packet from the receiver. In addition to the slow start and congestion avoidance mechanisms [75] described in Sec. 2.1.2.1 there has been various other proposals for adapting the window size to the network congestion state. To mention a few out of many:

Slow start and search (Tri-S): The Tri-S scheme [195] uses the changes in throughput as an indication of congestion. The algorithm computes the normalized throughput gradient and compares it to some thresholds to keep the connection at the optimal operating point.

Dual-Window: With this algorithm [196] the sender maintains two windows, one for congestion control and one for flow control. The flow window adapts the transmission rate of the sender in accordance with the available buffer at the receiver and the congestion window adapts the transmission behavior in accordance with the network congestion state.

Delay-based adaptation: Jain et al. present in [78] an approach for congestion control based on estimating the round trip delay. The sender estimates the round trip delay as the time passed between sending a packet and receiving an acknowledgment for it. The window size is then adjusted based on the variations in the measured round trip delay.

IFP: Jacobs [74] presents a scheme called the Internet-friendly protocol that uses the congestion control mechanisms of TCP, however, without retransmitting lost packets. In his scheme, the sender maintains a transmission window that is advanced based on the acknowledgments of the receiver, which sends an acknowledgment packet for each received one. For the case of losses the transmission window is reduced in a similar manner to TCP. Based on the size of the window the sender can estimate the appropriate transmission rate.

3.1.2.2 Rate-Based Adaptation

With window-based adaptation schemes the transmission rate of the sender is only controlled indirectly by adjusting the transmission window size. On the contrary, rate-based adaptation schemes control the rate of the sender directly. To mention just a few out of many schemes proposed in this area:

Scalable feedback adaptation: Bolot et al. present in [19] an approach for scalable control of multicast media streams. The senders probe the receivers to solicit feedback information about the losses observed at the receiving end systems. Based on the collected information the sender either additively increases its transmission rate or decreases it multiplicatively.

Backward congestion notification: Kanakia et al. present in [91] an approach for adapting the transmission of real-time video. The adaptation algorithm uses feedback information sent by the routers about the available buffers to determine the appropriate transmission rate.

Dynamic QoS control (DQoS): Busse et al. describe in [25] an approach that uses the loss and delay information carried in the RTP control messages as the basis for the adaptation decision. With DQoS the sender reduces its transmission rate by a multiplicative decrease factor after receiving an RTCP feedback from the receiver indicating losses above a certain threshold called the upper loss threshold (λ_c). With losses below a second threshold called the lower loss threshold (λ_u) the sender can increase its transmission rate additively. For the case that the feedback information indicates losses in between the two thresholds the sender can maintain its current transmission level.

RAP: Rejai et al. present in [146] a congestion control scheme called rate adaptation protocol (RAP). Just as with TCP, sent packets are acknowledged by the receivers with losses indicated either by gaps in the sequence numbers of the acknowledged packets or timeouts. Using the acknowledgment packets the sender can estimate the round trip delay. If no losses were detected, the sender can periodically increase its transmission rate additively as a function of the estimated round trip delay. After detecting a loss the rate is reduced by half in a similar manner to TCP.

TFRCP: Using the analytical model of TCP presented in Sec. 2.1.2.1 Padhye et al. [130] present a scheme in which the sender estimates the round trip delay and losses based on the receiver's acknowledgments. In case of losses, the sender restricts its transmission rate to the equivalent TCP rate calculated using Eqn. 2.6 otherwise the transmission rate is doubled. While the scheme behaves in a TCP-friendly manner during loss phases, its increase behavior during underload situations is rather arbitrarily chosen and might result in severe unfairness as the adapting end system might increase its transmission rate much faster than a competing TCP connection.

3.1.3 Congestion Control for Multicast Communication

In addition to the rough classification in window and rate-based congestion control in Sec. 3.1.2 for the case of multicast we can further classify the schemes based on the entity that actively changes the performance of the entire system. We divide the congestion control schemes for multicast communication into three main categories: sender-based adaptation schemes, receiver-based adaptation schemes and hybrid schemes.

3.1.3.1 Sender-Based Schemes for Multicast Congestion Control

Similar to the approaches presented in Sec. 3.1.2 the sender adapts its transmission behavior based on feedback information generated from its receivers. However, having each receiver frequently reporting feedback information would result in a feedback implosion at the sender. To avoid this situation, congestion control schemes for multicast communication usually use special mechanisms for reducing the flow of feedback information from the receivers to the sender and still provide the sender with enough information about the congestion state of the receivers. Such mechanisms can be roughly divided into the following categories [69]:

Distributed: In this case the receivers periodically transmit feedback information to the entire multicast group. Feedback implosion is avoided in this case by adjusting the length of the periods in accordance with the number of receivers. Such an approach is useful for small to medium group sizes. For the case of large groups of several thousand receivers the transmission periods get too large so that the feedback information arrive too infrequent at the sender to be used as the basis for an adaptation process. The control messages of the real time transport protocol (RTP) [158] and the session announcement protocol (SAP) [62] are examples for such an approach.

Suppression: Suppression of feedback messages indicates that a receiver suppresses the transmission of a feedback report if it has noticed that some other receiver has already sent a similar report. For example if a receiver wants to inform the sender about the loss of a data packet it schedules the transmission of a negative acknowledgment for the lost packet after some random time (T_r). After T_r the receiver multicasts its report to all other session members. If the receiver noticed a negative acknowledgment for the same packet form some other receiver before T_r expires it suppresses the transmission of its own report [124].

Representatives: While distributed feedback schemes do not scale well for large multicast groups, suppression based approaches offer the sender with loss or congestion information as seen by a subset of the receivers and not the entire group. To provide the sender with information representing the state of a large part of the receivers different approaches were proposed for collecting the feedback at intermediate nodes and presenting the sender with a summary of the gathered state information. Gao et. al. [59] and Lee et al. [101] describe approaches where the intermediate routers along the multicast router tree collect feedback information from the multicast leafs or nodes connected to them and sum up the information into a single report that is handed to a router higher in the multicast tree. Such approaches require changes in the functions and capabilities of the installed basis of multicast routers in the Internet. However, as such changes can only be realized after a longer transition period realizing such approaches is not feasible currently. Rhee et al. [147], DeLucia et al. [41] and Lin et al. [135] describe approaches with which only a subset of the receivers called representatives is allowed to send feedback information to the sender. This subset can be statically set as in [135] or dynamically chosen as in [41]. Each representative collects feedback information for a subset of the receivers and forwards the summarized information either directly to the sender or to a higher representative.

Polling: Bolot et al. [19] describe a feedback approach in which the sender polls feedback information form the receivers. This is done by having the senders and receivers generate a 16 bit random key. The sender sends a control message asking for the feedback with the generated key with all digits marked as significant. Only receivers with a similar key are allowed to send feedback information in this case. In subsequent rounds the number of significant digits transmitted in the control message is reduced and receivers with keys similar to the reduced key are allowed to send reports.

Hybrid: Different control schemes actually deploy a mixture of the here described feedback approaches. In [147], only representative nodes are allowed to send feedback information to the senders. However, in the single receiver subsets represented by a representative node suppression is used to collect information from the single leafs.

Similar to our discussion in Sec. 3.1.2 we can also for the case of multicast congestion control distinguish between window and rate-based adaptation schemes.

Window-Based Congestion Control As described in Sec. 3.1.2.1 window-based control schemes use a sliding window to control the amount of data the sender can transmit. This window is updated based on the feedback information from the receivers.

MTCP [147] and RMTP [135] use a congestion window similar to that of TCP with the main difference that the window is only increased if all receivers acknowledge the reception of some packet. The increase and decrease actions of those protocols are similar to that of TCP. As the receivers need to generate feedback information indicating the reception or loss of data packets a combination of suppression and representatives are used in this case to avoid feedback implosion.

Wang and Schwarz [194] describe a window-based congestion control scheme for multicast called random listening algorithm (RLA). Similar to MTCP or RMTP the receivers inform the sender about the lost and received data. The transmission window is only increased if a packet was acknowledged by all receivers. In case a loss was signaled the sender generates a uniform random number (τ) and reduces the window by half only if τ is larger than some threshold. Usually sender-based congestion control schemes adjust the transmission rate of the sender based on the worst loss and delay values measured by all receivers. Thereby the TCP-friendliness of the approach is determined by comparing the throughput of the multicast session on the path connecting the sender to the receiver that has measured the worst loss and delay values and the throughput of a TCP connection under similar loss and delay values (R_{TCP}). On the contrary, RLA bounds the transmission rate of the sender so that it lays between (R_{TCP}) and ($c \times R_{\text{TCP}}$) with c determined in accordance with the number of receivers in the multicast session.

The Illinoy reliable multicast architecture (IRMA), see Lee et al. [101], represents a different approach for emulating the TCP behavior for multicast communication. Instead of using UDP as the transport protocol IRMA proposes using a modified TCP that allows using multicast addresses. Thus, the sender and receivers connect themselves to IRMA multicast routers using TCP. Data packets are sent along the multicast tree and receivers send acknowledgment packets which traverse the same multicast tree but on the opposite direction. At each IRMA router the acknowledgments form the leaf nodes or the downstream routers are collected and summarized into one acknowledgment which is then sent upstream. As the sender uses TCP to realize the data transmission, the common TCP congestion control mechanisms control the transmission behavior of the sender.

Rate-Based Adaptation Rate-based adaptation schemes adjust the transmission rate of the sender directly based on the feedback information from the receiver.

In [59] Gao et al. present an approach called the rate congestion control scheme for multicast (RACOOM). In RACOOM the receivers send feedback information containing the sequence numbers of lost and received packets and a timestamp. Intermediate routers aggregate incoming feedback messages and send a feedback message to the upstream router indicating the worst loss and delay values indicated by the incoming feedback messages. Thus, the sender finally receives information about the congestion state of the path with the worst loss and delay values. The sender keeps an estimation of the round trip delay (T_{rtt}) towards its receivers as well as an estimation of the minimum round trip delay ($T_{\mathrm{rtt}_{min}}$) towards the receivers. After receiving acknowledgment packets indicating a no loss situation the sender uses $T_{\mathrm{rtt}_{min}}$ to determine the number of packets that should have been received by the end systems (N_{exp}) and compares this value to the actual number of received packets (N_{act}) as indicated by the information carried in the feedback messages. The sender then increases its transmission rate if ($N_{\mathrm{act}} - N_{\mathrm{exp}} < \alpha$) or decreases its transmission rate if ($N_{\mathrm{act}} - N_{\mathrm{exp}} > \beta$). In case of losses the sender reduces its transmission rate by half. The simulations presented in [59] suggest that the scheme's performance depends to a great extent on the values of α and β. As these values are set by the user the scheme needs further refinement to allow for dynamical setting of those values and allow thus for a more stable and predictable behavior.

With the monitor-based flow control (MBFC) [153] the sender polls its receivers by sending at periods of T specially marked packets. Upon receiving a marked packet the receivers start an observation period in which they count the number of incoming packets until another marked packet arrives. The receivers report the number of counted packets (N_{rec}) to the sender which compares this reported values to its locally maintained count of the sent packets (N_{send}). The sender determines the number of low loss receivers using the following inequity

$$\frac{N_{\mathrm{rec}}}{N_{\mathrm{send}}} > L_{\mathrm{low}}$$

and high loss receivers using the following inequity

$$\frac{N_{\mathrm{rec}}}{N_{\mathrm{send}}} > L_{\mathrm{high}}$$

Based on the determined number of low loss and high loss receivers the sender adjusts its transmission rate in a similar way to TCP. That is, the rate is reduced by half in case of congestion. For the case of a no loss situation the sending rate is increased by $\frac{T}{T_{\mathrm{rtt}}}$. Similar to RACOOM, MBFC uses static thresholds for determining the needed adaptation actions. Additionally, it does not specify the necessary approach to avoid feedback implosion on the one hand and get an estimation of the number of congested and non-congested receivers on the other.

Floyd et al. present in [61] a scheme in which the receivers estimate the rate the sender should be using with Eqn. 2.6 based on measured round trip delay and losses. For estimating the round trip time, the sender includes in its packets a timestamp indicating the transmission time of the packet. The receiver estimates the round-trip delay using some predefined default value or with the help of synchronized clocks as twice the time difference between the timestamp in the received packets and its local clock. The receivers then use a suppression mechanism for reducing the number of feedback messages sent back to the sender. As this approach relies solely on the TCP-model, the end systems always need to estimate some loss value. This is achieved based on the observations made over longer time intervals.

3.1.3.2 Receiver-Based Adaptation

In the adaptation approaches presented in Sec. 3.1.3.1 the sender actively adjusts its transmission rate to avoid overloading the links towards its receivers. However, due to the heterogeneity of the Internet links, the connections of the end systems to Internet and the wide variety of end systems with different processing and display capacities a single sender transmission rate cannot satisfy the conflicting bandwidth requirements at different sites. Therefore, the sender rate is usually adapted to the requirements of the worst positioned receiver, thereby reducing the quality of the data perceived at all receiving sites as in [61, 135, 59].

This limitation can be overcome using hierarchical layered transmission mechanisms. These mechanisms use a layered coder [114, 183] that exploits correlation across sub-flows to achieve better overall compression. The data stream is partitioned into a base layer, comprising the information needed to achieve the lowest quality representation and a number of enhancement layers. Each higher layer provides refinement information to the previous layers. The receiver must listen to all lower layers up to and including the highest one it wants to listen to, i.e., if a receiver wants to listen to layer 3 it also has to listen to layer 1 and layer 2.

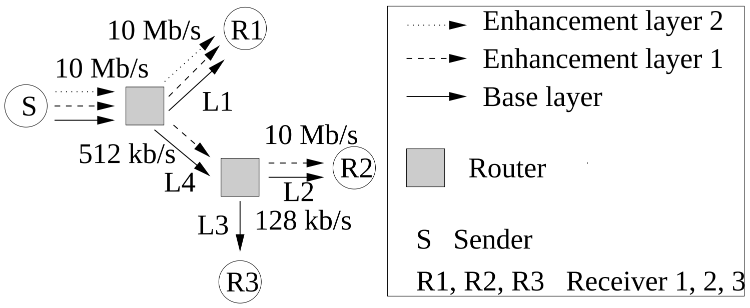


Figure 3.2: Example of multi-layer transmission

While MPEG-2 for example supports such a layering of data some problems might arise due to a drift problem caused by motion compensated coding if only the lower bit rate layer can be received. Under this condition, different reference frames for motion compensation are used at coder and decoder [71]. Another problem with cumulative layering is that of resynchronization. As the different layers might actually take different routes to the receiver, they might suffer from different delays. To reconstruct a data segment consisting of data packets sent over different layers the receiver needs to wait until all the different parts are received. This incurs additional delay and thus might reduce the overall perception quality.

Another approach for layered transmission is to simply transmit the same data stream encoded with different quality levels on different multicast sessions [104]. This scheme is often called *simulcast* because the source transmits multiple copies of the same signal simultaneously at different rates resulting in different qualities. As the streams contain all the necessary information for decompression, the receivers need only to join one multicast session. This approach avoids the resynchronization and drift problems seen with hierarchical approaches. However, this is reached at the cost of sending multiple replicated streams and thus possibly congesting the network. Due to this aspect we will restrict our discussion in this work to the hierarchical layering approach even though most of the presented work applies to data simulcast as well.

With hierarchical data transmission approaches, the different data layers are sent to different multicast sessions. All the receivers join the basic session and based on their requirements and capacities determine how many layers to receive. The congestion control is achieved here using the interaction of the receivers with the multicast routing protocol. The data of some multicast session is only forwarded to a multicast router X if at least one receiver connected directly or through a series of other routers to X has signaled its interest in receiving this multicast session using IGMP [39]. Thus if a receiver wants to reduce the rate of its incoming data stream it needs to send an IGMP message to its nearest multicast router (X_1) indicating it wants to leave the highest multicast layer. Upon this, X_1 checks if any other receivers or routers connected to it still want to receive this layer and if not informs the next upstream multicast router X_2 to prune this multicast layer so that the data of this layer does not get forwarded on the link connecting X_1 to X_2 and so on. Fig. 3.2 shows such a situation. A sender is transmitting its data in a base layer and two enhancement layers. Due to the limitation of its connection to the Internet receiver R3 only joins the base layer, which gets forwarded on his link. Receiver R2 has a faster Internet connection but has a bottleneck on the path between him and the sender. Thus, R2 subscribes only to the base layer and the first enhancement layer. Notice that these actions of joining and leaving the layers and the resulting selective forwarding of the data layers at the routers is not application specific and can be realized with the current Internet multicast infrastructure.

The basic approach for receiver-based adaptation using layered data transmission is for the sender to transmit its data in a number of hierarchically ordered layers. The receiver joins the basic layer and if no losses were observed it can join higher layers. After noticing some congestion the receiver needs to leave its highest joined layer. From our brief discussion above on multicast routing we can already identify two main problems with such a simple approach:

- Leaving a layer does only result in reducing the incoming data stream at some congested router (X) if all receivers connected directly or indirectly to X leave this session as well.

- A congestion state noticed after receiver R_x joins a higher layer (L_n) need not be caused by the join action of R_x but by receiver R_y joining layer $L_{\mathrm{n+m}}$. The join action of R_y can cause the forwarding of the data of layer $L_{\mathrm{n+m}}$ over all or part of the path between the sender and R_x and thus lead to the congestion situation.

To avoid these effects the adaptation scheme needs to deploy some kind of mechanisms to coordinate the actions of the different receivers.

In the following we present some approaches for realizing receiver-based adaptation:

RLM: With the receiver-driven layered multicast by McCanne et al. [113] each receiver drops its current highest joined layer if it measured losses. If no losses were noticed, the receiver can try to join a higher layer after some time period (T). If losses were measured after the join action the receiver leaves this layer and increases the value of T so as to reduce the rate at which it tries to join this layer in the future. Otherwise the layer is kept. Scalability and interference problems introduced with the join experiments are solved by having the distributed receivers learn form each other about the failed join actions.

LVMR: Li et al. describe in [103] a scheme called layered video multicast with retransmission (LVMR). The architecture of LVMR is based on using multiple domains within a multicast group each represented by an intermediate agent (IA). A domain can be a geographical area or a corporate network. Each domain is then further divided into subnets represented by a subnet agent (SA). SAs collect information about the success or failure of the join actions of the receivers in their subnet. This information is propagated to other members of this subnet as well as to the IA responsible for this SA. The IAs collect information from their SAs and propagate them to their parent IA as well as to the SAs in their domain. The tight cooperation between the SAs and IAs reduces the possibility of failed join actions, however, at the cost of the higher complexity required for building the logical tree of SAs and IAs and the additional control messages.

Thin streams: Instead of using a few distinct layers as was discussed in RLM [113], Wu et al. [200] propose to divide a video stream into several thin layers with only little data. Using this approach allows for more granular adaptation and reduces the effects of failed join experiments. That is, if joining a new layer causes the amount of transported data over some link to become larger than the available resources, the congestion level will only be small as joining a new layer results only in a slight increase in the overall amount of forwarded data. Receivers measure the rate of the incoming data stream and compare it to the expected value of the subscribed layers. A difference larger than some threshold indicates a congestion situation and results in dropping the highest layer. The join actions of the receivers are synchronized by sending a clock pulse over the base layer.

RLC: With the receiver-driven layered control scheme Vicisano et al. [191] describe a scheme in which the receivers join or leave a layer based on their measured loss values. Using specially flagged packets the sender indicates synchronization points at which receivers might join or leave a specific layer.

TDLM: There are a row of schemes we call TCP-driven layered multicast [182, 187] that similar to [107] use Eqn. 2.5 for determining the appropriate bandwidth share a receiver can utilize. The sender transmits its data in layers and the receiver joins then a number of layers that have an accumulated bandwidth close to its determined bandwidth share. The schemes [182, 187] are, however, still early studies as they do not discuss how to accurately determine this TCP-like share at the receivers without knowledge of the round trip delay and how the join and leave actions should be conducted.

PLM: The packet-pair receiver-driven layered multicast (PLM) [102] is based on layered transmission and on the use of the packet-pair approach to infer the bandwidth available at the bottleneck to decide which are the appropriate layers to join. To apply the packet-pair approach for reliably estimating a flow's bandwidth share, the authors of PLM assume that all router in the network deploy some kind of a fair queuing mechanisms that allocate each flow a fair bandwidth share.

With PLM the sender periodically sends a pair of its data packets as a burst to infer the bandwidth share of the flow. The receivers use the gaps between the specially marked packet-pairs to estimate their bandwidth share. Based on the estimated share they determine the number of data layers they can join.

3.1.3.3 Hybrid Adaptation Schemes

With layered transmission approaches, a data stream is usually divided among different layers based on encoding characteristics or possible network topologies. For example to support users connected through ISDN, T1 and ATM links to a multicast session, a sender might divide its data stream into three layers with the base layer having a data rate of 64 kb/s, a first enhancement layer with a few hundreds kilobits per second and the rest of the data to be sent is then placed on the second enhancement layer. However, with no a priori knowledge of the traversed links, such a static distribution might be disadvantageous as it does not consider the actual resources of the receivers. For example, consider the distribution presented above for accommodating receivers connected over ISDN, T1 and ATM links. For the case that no members are connected over ISDN links, and some members have enough resources to receive a data stream larger than the base layer but smaller than the first enhancement layer, they would still be restricted to the lower layer.

To better accommodate the actual heterogeneity of the network and receivers some interaction between the senders and receivers is required. In [6] Amir et al. describe a mechanism called SCUBA that enhances layered data transmission schemes by assignment of signal layers to network channels based on user preferences computed by the SCUBA scheme. Given a user preference, each source determines its signal layer mapping. While this approach accommodates receiver preferences in accordance with the sent data, the QoS of the participating receivers is still only changed with the granularity of a layer and provides no assistance in dynamically determining the appropriate transmission rate of each layer.

Albuquerque et al. [4] describe an approach called source adaptive multilayered multicast (SAMM), in which the receivers inform the sender about their desired transmission rate. To avoid feedback implosion, the feedback messages are not sent directly to the sender but to a representative that aggregates the received data in the form of a vector of the rates requested by the receivers and the number of receivers requesting each rate. The feedback messages arriving at the receiver, display then the actual heterogeneity of the network and contain the number of layers the sender should be using and the size of each layer. The receivers estimate the appropriate transmission rate for them by measuring the losses of the incoming data stream.

Jiang et al. [86] present a scheme in which the sender transmits its data in a fixed base layer and a variable enhancement layer. Based on their measured losses, the receivers estimate their bandwidth share and report this to the sender. The transmission rate of the variable layer is then determined as to increase the satisfaction of most of the receivers.

3.1.4 Network-Based Adaptation

Aside form various proposals for increasing the fault-tolerance of networks [123, 93] and adapting the transmission of routing information in accordance with network congestion [126] various studies have investigated network-based mechanisms for adapting multimedia streams in accordance with the network congestion state. Placing the adaptation entities in the network has the clear advantage of solving the congestion problem at the point of happening. However, such an approach requires a network capable of actively reshaping the transfered data which adds a considerable load on the network nodes and increases the complexity and costs of building and operating such networks.

The proposals for network-based adaptation schemes range form simple buffer management approaches up to complicated QoS architectures. To describe just a few examples:

Available Bit Rate (ABR): The ABR service [162] was specified as an ATM service for applications mainly handling data transfer and having the ability to reduce their sending rate if the network requires them to do so. Likewise, such applications may wish to increase their sending rate if there is extra bandwidth available within the network. Applications using the ABR service can expect the following quality of service commitments from the network:

1. The available bandwidth is fairly distributed among all active ABR connections.
2. A minimum cell rate that is agreed upon during the connection establishment phase.
3. Only a preset fraction of the sent data can be dropped as long as the sending behavior of the application conforms to the negotiated values.

With the ABR service a source sends a so called resource management (RM) cell every N sent data cells. These cells indicate the current cell rate (CCR) and the desired one of the senders. The destination end system turns the RM cells around and sends them back to the source. The intermediate switches determine the fair bandwidth shares the ABR connections should use in order to avoid congestion based on the traffic situation in the network. These values are then written into the backward RM cells in the explicit rate (ER) field. The source should then increase or decrease its rate in accordance with the explicit rate noted in the RM cells. Additionally, the network switches could also just indicate their congestion situation by a single bit indicating an over- or underload situation.

Early congestion notification (ECN): TCP relies primarily on detecting loss events for activating its congestion avoidance mechanisms. That is, a TCP sender only reacts to the congestion situation when it is already too late and packets have been lost. Floyd et al. [52] present an approach in which the network nodes indicate their congestion situation by setting a congestion bit in the TCP packets. As a guideline for the sender's reaction to the congestion bits Floyd et al. propose the following:

1. TCP's response to ECN should be similar over longer time scales to its response to dropped packets or timeouts.
2. Over smaller time scales TCP's response to ECN can be less conservative than its response to lost packets.
3. TCP should react to ECN at most once per round trip time.
4. The response to ECN does not trigger the sending or retransmitting of any data packets.

With ECN, TCP senders can reduce their transmission windows before packets get dropped and thereby avoid the long recovery periods caused by data losses.

In a related approach, Sisalem et al. [170, 171] present a scheme called binary congestion notification (BCN) also based on congestion bits. However, the sender behavior is adjusted in such a manner as to ease the integration of TCP and ABR. With BCN, receiving an acknowledgment with the congestion bit set does not result in reducing the transmission window by half as with ECN but by a multiplicative decrease factor as is the case for ABR. This allows for a smooth integration of ABR and TCP.

Explicit windows: For a better integration of TCP and ATM, Kalampokas et al. propose in [90] to adjust the window information carried in the TCP acknowledgment packets based on the available bandwidth share calculated by the network nodes. This approach removes the need for the additive increase and multiplicative decrease used by TCP for setting the transmission window size and, hence, leads to a better overall performance.

PLoP: The predictive loss pattern queue management scheme [152] proposes a mechanism for reducing loss burstiness of audio streams and thus allowing for more efficient usage of forward error correction approaches .

Video gateway: In an effort to reduce the bandwidth of a stream to better suite the capacities of a receiver Amir et al. propose in [7] using gateways for transcoding high bandwidth video streams into low bandwidth ones. These gateways can be placed at points connecting high and low capacity links or where congestion is very likely to happen.

Filtering: In order to overcome congestion situations and reduce the effects of packet losses on the perceptual QoS of MPEG video streams, Yeaden et al. [202] propose a network based mechanism for discarding complete frames that are not essential for reconstructing the original video stream. Thereby, the amount of data transported in the network can be reduced and congestion avoided.

Active networks: Active networks [184] also present a realization possibility of network-based adaptation. With such an approach a network node can reshape a data stream based on information and procedures provided by an entity with knowledge about the data stream.

Throttling: Floyd et al. [56] describe a mechanism in which routers identify flows that use more bandwidth than allowed by Eqn. 2.5 and then throttle these flows.

Those are just a few of many approaches, see also the IPCSS [140], KISS [87], QoS-A [26] proposals for making networks more adaptive and enhancing the network's support for multimedia communication for a wider range of users.

3.2 Quality of Service Control for Multimedia Communication

The packet switching architecture of the Internet provides a flexible infrastructure for the interconnection of networks with different architectures. However, it provides only for a best effort service with little control over the packet delay and loss processes at the network nodes. This best effort service makes it impossible to provide guarantees for a minimum bandwidth or a maximum delay.

One approach for providing an improved QoS control in the Internet is to augment the current best effort service to include new services that provide performance guarantees desired by the applications and users. In this section, we briefly describe a few of the most current proposals for such enhanced services.

3.2.1 The Integrated Services Model (IntServ)

The integrated services framework [22] provides the ability for applications to choose among multiple, controlled levels of delivery service for their data packets. To support this capability, two things are required:

1. The individual network elements such as the subnets and IP routers along the path traversed by an application's data packets must support mechanisms to control the quality of service delivered to those packets. These mechanisms can be roughly divided into three parts:

   Call admission control: This instance checks if the requested reservation level can be supported by the router [85].

   Classification: In order to determine the appropriate service to provide for a data packet belonging to some flow this packet needs first to be classified based on the previously made reservation for this flow and is then inserted into the appropriate scheduling level.

   Scheduling: This instance determines when to forward a data packet based on the reservation made for it.

2. A way to communicate the application's requirements to the network elements along the path and to convey QoS management information between the network elements and the application must be provided.

The integrated services model supports three service types:

Guaranteed load: Maximum delay and bandwidth requested by the application are guaranteed [160].

Controlled load: Only a minimum bandwidth is guaranteed. Traffic sent above this minimal rate is treated as best effort traffic [199].

Best effort: Data packets are sent without any guarantees.

For conveying the application's requirements the IETF has defined the resource reservation protocol (RSVP) [21]. With the resource reservation protocol (RSVP) Braden et al. have proposed a QoS signaling protocol, in which the sender informs the receivers about the traffic shape of the data flow. In reaction to that, the receivers send reservation requests indicating the amount of required resources to support the flow.

Each network node traversed by the control messages maintains information about the requested resources for each flow. Reservation requests by different receivers for the same data flow are merged together at the network nodes. For each flow that has made a reservation, the network nodes periodically send control messages to their neighboring nodes to indicate the state of this flow. As all the nodes need to maintain state information for each flow and send and receive control messages to and from neighboring nodes to detect the state of each flow, RSVP incurs a significant processing overhead on the network nodes. In addition, various problems resulting from issues like merging of reservations and reservation updates need to be considered with RSVP.

Pang and Schulzrinne propose a sender based reservation scheme called YESSIR [131] in which the sender issues the reservation request. Each traversed router processes the reservation request, decides to accept or reject it and establishes some state information for the signaling flow in case of acceptance. After receiving a reservation message from the sender, the receivers issue acknowledgment packets indicating the result of the reservation. After establishing a reservation, the sender periodically transmits control messages to refresh its state at the network nodes. This approach reduces the complexity of RSVP as it does not require the routers to exchange per-flow refresh messages. However, it still requires per-flow state information at the routers.

3.2.2 The Differentiated Services Model (DiffServ)

The need for maintaining per-flow states, periodically refreshing this information as well as the overhead of per-packet classification and processing results in scaling problems for the IntServ model and RSVP when handling a large number of flows. The differentiated services (DiffServ) model [14] was designed to avoid this problem by using service level agreements (SLA) between the network providers and the users. These SLAs describe the QoS level the aggregated traffic of a user can expect from the provider. Traffic sent in conformance with the established SLA is marked as belonging to a specific QoS level. At the core routers data packets are serviced differently based on their marked QoS level. As all packets with the same marks are treated equally, the core routers need only to maintain information describing the resources allocated for the supported QoS levels.

As the SLAs are used for traffic aggregates, there is no need for per-flow states. Additionally, with the DiffServ model only the edge routers need to police incoming traffic and take admission control procedures.

Broadly speaking, any traffic management or bandwidth control mechanism that treats different users differently -ranging from simple weighted fair queuing (WFQ) [33] to RSVP and per-session traffic scheduling- counts. However, in common Internet usage the term is coming to mean any relatively simple, lightweight mechanism that does not depend entirely on per-flow resource reservations.

While such an approach avoids the scalability problems of the integrated services model it is rather rigid. SLAs are currently mainly thought to be established in a static manner or to be changed only infrequently on the order of days or weeks. Hence, the DiffServ model does not allow the user to increase or decrease the amount of its reserved resources in accordance with its traffic requirements. In addition to the static nature of the SLAs specifying a QoS level for an aggregate of flows can result in unfair distribution of resources among the flows belonging to the same aggregate due to the aggressiveness of some flows, differences in round trip delays and taken paths[122]. Further, the core routers are expected to be dimensioned large enough in a manner as to provide the user with the agreed upon QoS level on any path taken by the user at any time in accordance with the SLA. As the exact path taken by a user's traffic is not known in advance the network needs to be highly over provisioned to account for all possible cases. This is even more pronounced for the case of multicast as the user's traffic might take different paths in the providers network and hence enough resources need to be available on all paths simultaneously. To allow for dynamical establishment and changes of the resources provided for a user there are currently different proposals that aim at introducing light-weight signaling protocols [168] as well as different efforts for aggregating state information [12] to reduce the load on the network routers.

3.2.3 QoS Architectures

In addition to the above models, various studies [50, 8] describe complete architectures for reserving network resources and controlling the QoS of the network traffic. While these studies offer a good basis for the QoS research and first experiences, they are too complicated and could, thus, not gain any significance in the Internet.

While these approaches differ in their scope and complexity they all have common requirements:

- The networking infrastructure needs to be substantially enhanced in order to realize the improved services. This involves improving the network routers by introducing more complicated buffer management schemes than simple FIFO for enforcing the requested QoS levels, admission control mechanisms to effectively utilize the network resources as well defining signaling mechanisms for establishing the required QoS by the users.

- Provide incentives to encourage applications to request the proper service class for their requirements. In the absence of such incentives, applications could request the highest quality level no matter what their requirements are. One way for providing incentives is via pricing the communication service based on the requested QoS.

- In order for the users and applications to benefit from the upgraded networking infrastructure the applications need to be able to specify their requirements and support the appropriate signaling protocols for doing so.

3.3 Adaptation versus Reservation

While reservation based schemes guarantee the required QoS level, deploying such mechanisms inclines a substantial increase in the complexity of the network due to the necessary enhancements of the routers and introduction of billing mechanisms [16]. Additionally, due to the admission control, the user satisfaction of the network service might be reduced due to the possibility of blocking a service request during overload periods.

On the contrary, end-to-end congestion control schemes require no or only minimal enhancements to the network and rely, mainly, on enhanced end systems. Such adaptive mechanisms aim at improving the quality by adjusting the amount of traffic in the network and hence reducing losses on the one side and increasing utilization on the other which is particularly beneficial for multimedia communication. So, at the expense of slight degradation in user satisfaction, it is possible to adjust the amount of network resources consumed by a media stream in accordance with the network congestion state. Additionally, it is usually the case that due to the loss of content or the need to use a considerable overhead for forward error correction a low bandwidth video stream, for example, with no or only low losses can have a higher perceived quality than a high bandwidth yet lossy stream [189]. Further, as the adaptive applications are better suited to take advantage of dynamical changes in the network resource availability during a session, which is particularly beneficial during long-lived sessions typical for multimedia communications, higher network utilization levels can be reached. Such a behavior can be achieved by reservation-based schemes only by renegotiating the reservation [60] which further increases the complexity of these schemes.

In an analytical study, Breslau et al. show in [24] that by over-provisioning the capacity of a network, a best effort network can provide the same quality levels as a reservation-based network. Thus, for providing a network supporting a high QoS there is a clear trade off between the costs of introducing reservation-based schemes and the thereby required network enhancements, billing mechanism and admission control mechanisms on the one hand and the costs of additional network resources required for a best effort network in order to provide the same QoS level [125]. For both cases, the costs are considerably decreased when deploying adaptive applications. For the reservation case, the applications require a smaller amount of network resources and less stringent control mechanisms. For the best effort case, users' satisfaction with the QoS can still be high even with a smaller amount of resources and thus allowing for less over-provisioning of the resources.

Whether the future high quality networks will be based on strict QoS control mechanisms or simply be over-provisioned will be a question of costs. In any case, efficient adaptation mechanisms will be required to reduce the overall costs.

As a combination between the two approaches, one might consider integrating adaptation schemes with a reservation scheme such as YESSIR [131] or RSVP. Through this integration, the receiver can reserve a minimal QoS using YESSIR and try to achieve a better QoS level through utilizing free network resources using adaptation. Alternatively, an approach similar to the scalable resource reservation protocol (SRP) [5] or renegotiated constant bit rate (RCBR) [60] can be integrated with adaptation. That is, the sender keeps informing the network about its desired transmission rate and the routers indicate the actual reserved capacity. Additionally, the senders can still try to use a higher transmission rate using an adaptation scheme.

Another important issue to consider is the aspect of flow protection. That is, protecting adaptive flows from unresponsive ones that do not reduce their transmission rate during congestion states and might thus lead to the starvation of adaptive traffic. Without protection schemes at the routers, a greedy connection can cause the adaptive flows to face losses and thus reduce their bandwidth share. Lakshman et al. [180] present an intelligent buffer management approach that supports the isolation of responsive from unresponsive flows in a simple way.

Chapter 4

TCP-Friendly Congestion Control for Point-to-Point Communication

In order to avoid overloading the network, sending end systems need to adapt their transmission rate in accordance with the available resources in the network. This adaptation process needs to be designed in a manner as to increase the network utilization, reduce losses and allow for a fair bandwidth distribution among competing flows.

Without exact knowledge about the resource availability in the network the end systems need to adapt their transmission behavior using some variation of the decrease/increase (ID) approach. That is, by increasing the transmission rate during underload situations and reducing it otherwise. In the end-to-end communication scenario discussed in this chapter, the end systems can not rely on the network routers for information. Hence, to inform the sender about the congestion state in the network, the receiver collects information about the losses and delays as measured form the incoming data stream from the sender and sends this information back to the sender. Instead of introducing a new control protocol for collecting and delivering loss and delay information, in this study we will rely on the real time transport protocol (RTP), see Sec. 2.1.3.1.

In Sec. 4.1, we first describe some of the implications and advantages of using RTP for collecting and transporting feedback information about the losses and delays in the network. In Sec. 4.2, we shortly discuss the performance of adaptation schemes that only consider loss and utilization aspects and do not take TCP-friendliness into account. In Sec. 4.3, we present an adaptation scheme called the loss-delay based adjustment algorithm (LDA) that while still using the infrequent RTCP control information manages to be TCP-friendly. The performance of LDA is then investigated using simulations and measurements in Sec. 4.4.

4.1 Implications of the Feedback Mechanism on the Design of the Congestion Control Scheme

By using RTP for establishing a closed control loop between the sender and the receiver, the sender can use the loss and delay information in the RTCP messages sent by the receivers as the decision basis for the adaptation algorithm.

Notice, however, that using RTCP as the feedback mechanism not only imposes novel challenges on the design of adaptation algorithms but also limits the achievable benefits gained using adaptive end systems. Most of the algorithms presented in Sec. 3.1.2 use feedback information arriving on short time scales of only a few milliseconds or a feedback message for each sent packet. Hence, the sender is able to react to rapid changes in the network. With RTCP, however, the senders receive feedback information only on the scale of a few seconds. The infrequency of the feedback messages dictates that such an approach can not benefit fast enough from rapid changes in the network conditions. Thus, the goal of RTCP based adaptation is to adjust the sender's transmission rate to the average available bandwidth and not react to rapid changes in the buffer length of the routers for example. This might be actually more appropriate in some cases than rapidly changing the transmission rate at a high frequency. For example, it is often the case that a video stream with a stable yet low frame rate is more acceptable for the user than a rapidly changing frame rate even though a high rate can be achieved for short periods. Additionally, multimedia applications are usually robust against low losses in contrast to data applications.

In terms of algorithm design the sender can not use a window-based adaptation approach. Taking the example of TCP, for a window based scheme to work efficiently it needs to use a maximum window ($W_{\max}$) equal to the size of the available bandwidth multiplied with the round trip delay [75]. In this case, the round trip delay indicates the time passed between sending a packet and receiving an acknowledgment for it. In the steady state, the sender would be able to send a ($W_{\max}$) worth of packets before waiting for an acknowledgment. Afterwards each arriving acknowledgment would trigger the sending of a number of packets equal to the acknowledged number of packets.

Using RTCP, the delay between sending a packet and receiving some kind of acknowledgment for it does not only consist of the round trip propagation delay and the buffering delays at the routers but consists mainly of the time interval between the transmission of two subsequent RTCP messages by the receiver. As this delay can be in the order of several seconds $W_{\max}$ would assume a very large value. Also, as the RTCP messages usually acknowledge the reception of a large number of packets[1] the sender would be allowed to insert a large burst of packets into the network, which might lead to losses. Based on these observations rate-based adaptation is the preferred adaptation approach when using infrequent feedback messages as is the case here.

4.2 Pitfalls of RTP-Based Adaptation Schemes

To demonstrate some of the problems that might arise when designing an adaptation algorithm without paying close attention to the aspects of fairness, we investigate in this section an approach by Busse et al. [25] called dynamic QoS control (DQoS). DQoS has great resemblance to a variety of other schemes described in [19, 165, 44]. With DQoS the sender increases its transmission rate by a constant additive rate whenever the loss value received from the receiver is below a certain threshold (λ_c). Whenever the loss value is above another threshold (λ_u) the sender reduces its transmission rate by a multiplicative

[1] Actually, the RTCP messages only indicate the highest sequence number seen at the receiver.

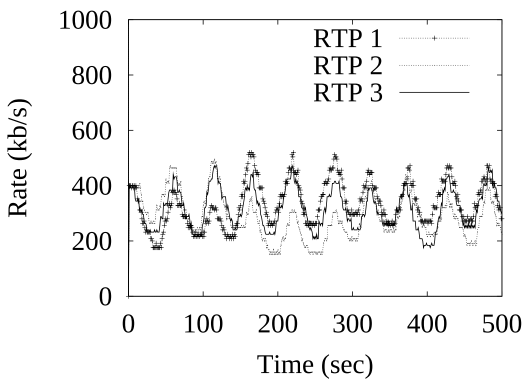
factor. The sender maintains its current transmission rate in case the losses are in between the two thresholds. In order to obtain a smooth loss indication the loss values reported by the receiver are filtered using an exponential moving average. DQoS uses RTCP for establishing the control loop between the sender and the receiver.

To test the performance of DQoS in terms of network utilization and fairness we simulated a simple network topology consisting of three DQoS flows ($m = 3, n = k = 0$) sharing a bottleneck of bandwidth (R) of 1 Mb/sec with a round trip time (τ) of 10 msec and a buffer size of 30 packets, see Fig. 2.5. The router is a random early drop (RED) gateway as was proposed by Floyd and Jacobson [55] which has the major effect here of avoiding synchronization and ensuring that all flows have a similar packet loss probability.

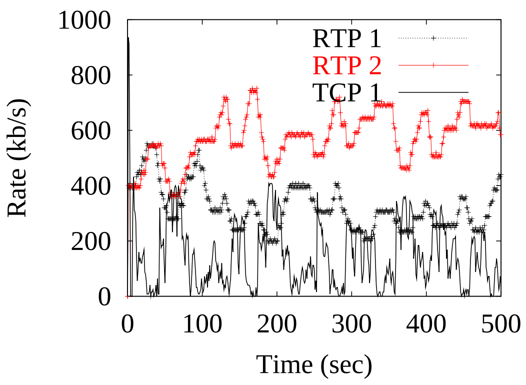
In Fig. 4.1(a) all senders are RTP senders using the DQoS algorithm with the minimum loss threshold (λ_c) set to 2% and the maximum one (λ_u) to 4%. The filter parameter was set to 0.3, the additive increase rate to 50 kb/s and the multiplicative decrease factor to 0.875. Those values were suggested in [25] to be most appropriate based on measurements.

The results depicted in Fig. 4.1(a) show that the scheme achieves around 90% bandwidth utilization with all the flows receiving similar bandwidth shares.

Fig. 4.1(b) shows the results of using the same test topology but with two RTP flows competing with a TCP connection ($n = 1, m = 2, k = 0$). While still having a high bandwidth utilization of more than 90%, the bandwidth is, however, unequally distributed among the flows. Actually, the TCP connection receives on the average less than 15% of the available bandwidth. Based on the fairness definition of Sec. 3.1.1.4 we would expect the TCP connection to receive around 33% of the available bandwidth. We also notice that in this test the bandwidth is not equally distributed among the two RTP flows.



(a) RTP senders



(b) RTP and TCP senders

Figure 4.1: Bandwidth distribution using DQoS

As the increase and decrease phases of the scheme do not consider the fairness issues towards TCP connections, the observed unequal bandwidth distribution was to be anticipated with this scheme. The used increase and decrease parameters were adjusted for the tested environment and would probably lead to different results for different test cases.

Additionally, choosing a minimum acceptable loss value contradicts the behavior of TCP which reacts to every loss indication. Hence, we see that the TCP flow in Fig. 4.1(b) is nearly starved compared to the DQoS flows.

4.3 The Loss-Delay Based Adjustment Algorithm - (LDA)

The loss-delay based adjustment algorithm (LDA) is a sender-based adaptation scheme, that adjusts the transmission rate of UDP-based multimedia flows to the congestion situation in the network in a TCP-friendly manner. Basically, LDA regulates the transmission rate of a sender based on end-to-end feedback information about losses, delays and the bandwidth capacity measured by the receiver. With no observed losses, the sender can increase its transmission rate additively otherwise it needs to reduce it multiplicatively. The increase and decrease behavior should result on the average in a TCP-friendly bandwidth share. That is the sender should not increase its transmission rate during a loss free period faster than a TCP connection. During loss periods the sender needs not to reduce its transmission rate lower than an equivalent TCP connection as indicated by Eqn. 2.6.

In Sec. 4.3.1 the issue of determining and transporting information about the network congestion state are presented. In Sec. 4.3.2 various possible approaches for realizing additive increase and multiplicative decrease rate control are investigated in terms of their TCP-friendliness and flexibility. Due to the restrictions of most of the investigated schemes we present novel ways to be used with LDA for increasing and decreasing the transmission rate in a flexible and TCP-friendly manner.

4.3.1 Measurement of Path Characteristics

From Eqn. 2.6 it is obvious that for determining a TCP-friendly bandwidth share losses as well as delays on the links between the sender and the receiver need to be taken into account. Additionally, the sender should not increase its transmission rate above the bottleneck rate of the link, i.e., the bandwidth of the smallest router on the path connecting the sender to the receiver.

LDA uses the real time control protocol (RTCP) for transporting control information between the sender and the receiver. RTCP messages already include information about the losses and delays noticed in the network, see Sec. 2.1.3.1.

In addition, we enhanced RTP with the ability to estimate the bottleneck bandwidth of a flow based on the packet-pair approach described by Bolot [15], see Sec. 2.2.3.

We added to the RTCP packets an application defined part (APP) including the source sequence number, the sequence number (SEQ) of a data packet that will start a stream of probe packets and the number (n) of probe packets that will be sent. Then, n data packets starting with packet numbered SEQ are sent at the access speed of the end system. At the receiver side, the bottleneck bandwidth (R) is calculated as:

$$R = \frac{\text{probe packet size}}{\text{gap between 2 probe packets}} \tag{4.1}$$

Note, that by using data packets as probe packets the additional bandwidth required for the bottleneck probing is restricted to the increased size of the RTCP packets. Sending data packets as probe packets is appropriate when transmitting video streams as one can send a few packets belonging to the same video frame together without considerably altering the traffic characteristics. Also, video packets tend to be large which increases their probability of being queued at the routers. For the case of audio streams with small packets and stringent timing requirements another solution might need to be considered.

4.3.2 Investigation of Design Options for Specifying Adaptation Schemes

Without detailed information about the bandwidth share a data flow can consume in the network, congestion control schemes usually refer to probing the network. That is, during underload situations indicated by low losses or delays, the sender of a flow can gradually increase its transmission rate. After a loss or delay indication the sender needs to reduce it bandwidth share again.

In the context of TCP-friendly congestion control, the increase and decrease actions of a sender should not result in the starvation of competing TCP connections. Further, in the context of multimedia communication, the increase and decrease actions should be designed in a manner as to allow for smooth and stable reactions to the network congestion state without leading to an oscillatory behavior that reduces the perceived QoS by the user. Additionally, as congestion control schemes need to operate over a wide variety of networks with different loss, delay and bandwidth values, the increase and decrease actions should be dynamically adapted to the environment they are used in.

In Sec. 4.3.2.1 and Sec 4.3.2.2, we present some of the proposals for designing the increase and decrease phases of control algorithms and investigate their performance in terms of TCP-friendliness, stability and applicability in different environments. Additionally, we specify novel approaches that allow for TCP-friendly congestion control in different networks with different parameters.

4.3.2.1 Options for Designing the Increase Phase

The increase and decrease phases are the major blocks influencing an algorithm's performance in terms of bandwidth utilization, loss ratio and fairness.

To achieve a high and stable perceived QoS at the end systems the resource share some end system is utilizing needs to converge to a stable value. However, with no information about the explicit share to use, increase/decrease (ID) schemes do not converge to a single steady state. Instead, the system reaches an equilibrium (steady state) in which it oscillates around the optimal state [34]. The time it takes to reach this equilibrium (responsiveness or transient state) and the size of the oscillations (smoothness) jointly determine the convergence of the scheme. Ideally, the time as well as the oscillation should be small, see Fig. 4.2 [34].

Constant Additive Increase Rate

With this approach the sender increases its transmission rate during underload periods by

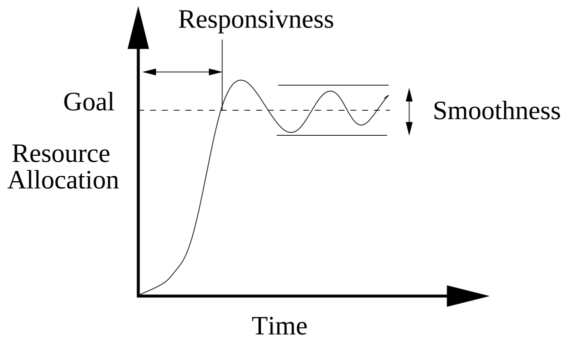


Figure 4.2: Responsiveness and smoothness of increase/decrease schemes

a constant amount. While this approach is rather simple to implement its performance in terms of smoothness and responsiveness depends to a great extent on the chosen increase value.

For the case of a single end system trying to utilize the maximum possible resource share (X_{goal}), the number of observation intervals (n) required for achieving the (X_{goal}) is

$$n = \frac{X_{\text{goal}} - X_0}{R_{\text{ai}}} \tag{4.2}$$

with X_0 as the initial resource share of the system and R_{ai} as the additive increase rate.

After converging to X_{goal} there will be a maximum overshoot (o) of

$$o = |R_{\text{ai}}| \tag{4.3}$$

That is, o indicates the transmission rate above X_{goal}.

From Eqn. 4.2 and Eqn. 4.3 it is evident that choosing a large value of R_{ai} results in a short transient period but also in a decrease in the smoothness and in higher oscillations [34] due to the increased overshoot.

Constant increase rate values have actually been successfully used in various schemes [78, 148]. However, these schemes adapt the transmission rate of the sender in short intervals in the range of a few milliseconds [148] or are even triggered by acknowledgment packets sent in response to each transmitted one [78]. This allows using a small additive increase rate which results in a good degree of smoothness. As the adaptation process is triggered on short time intervals the responsiveness of the system does not suffer from using small increase values. However, such an approach requires using a special protocol and the acknowledgments introduce additional load to the network. Using RTCP for collecting the information about the congestion state in the network, the receivers send RTCP packets around every five seconds. Thus, using a small R_{ai} results in long transient periods. On the other hand, using a large value would result in a reduced smoothness with oscillations of high amplitudes which can result in high packet losses during the overshoot periods.

An additional problem here is the appropriate definition of small and large R_{ai} values. Depending on the available resources and the share a flow can use, the appropriate value of

R_{ai} might change. For example, consider the case of a 1 Mb/s link shared between 2 flows. An ideal sharing would be if each flow received 500 kb/s. For this case, measurements conducted in [25] suggest that an appropriate R_{ai} value that allows small convergence periods and only small oscillations would be 50 kb/s. This value would, however, be inappropriate if the link was being shared among 100 flows for example or the link capacity was only 64 kb/s.

Dynamic Additive Increase Rate

To Avoid the problems of using a constant additive increase rate and to adapt the behavior of an end system in accordance with the available resources, the senders need to dynamically choose the appropriate R_{ai} value [17].

To achieve short transient periods, a high degree of smoothness and allow for fair bandwidth distribution the mechanisms for determining the additive increase rate need to

- lead to a large rate increase during the transient period,
- only result in a small rate variations during the steady state,
- allow flows with a lower bandwidth share to faster increase their shares than flows with already a high share,
- not to allow the sender to increase its transmission rate higher than the bottleneck rate of the path connecting the sender and the receiver and
- not to lead to a share increase more than a competing TCP connection would grab under similar loss and delay conditions.

Instead of taking constant values for the increase value, we propose in the study a novel solution that fulfills these requirements by scaling the increase rate in an inverse relation to the already acquired bandwidth share compared to the maximum possible one. The maximum possible bandwidth share indicates the smallest bandwidth of all routers on the path traversed by the flow. This value, termed bottleneck bandwidth value can be measured using the packet-pair probing approach as was described in Sec. 4.3.1. For all the simulations conducted here and the analysis we will consider the bottleneck value (R) to be fixed and a priory known to the sender.

We consider here two possible algorithms for realizing such a relation, linear and exponential relation. For a maximal possible bandwidth share R, i.e., bottleneck bandwidth, and a current transmission rate of (r_{i}) the sender can increase its transmission rate linearly by

$$R_{\text{ai}_i} = (1 - \frac{r_{\text{i-1}}}{R}) \times r_{\text{i-1}} \tag{4.4}$$

or exponentially by

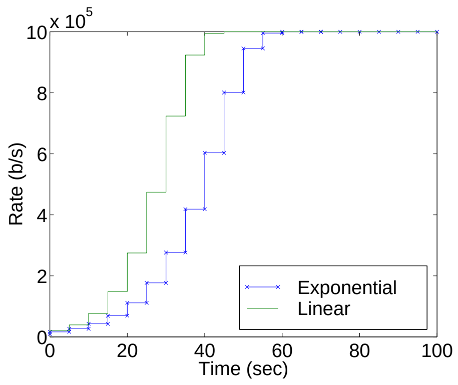
$$R_{\text{ai}_i} = (1 - \exp^{-(1-\frac{r_{\text{i-1}}}{R})}) \times r_{\text{i-1}} \tag{4.5}$$

and set its transmission rate to

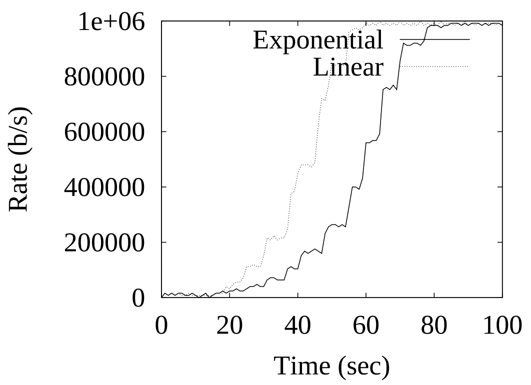
$$r_{\mathrm{i}} = r_{\mathrm{i\text{-}1}} + R_{\mathrm{ai}_i} \tag{4.6}$$

Fig. 4.3(a) depicts the adaptive behavior of a sender increasing its transmission rate using Eqn. 4.4 and Eqn. 4.5 in intervals of 5 seconds. The sender starts with a transmission rate of 10 kb/s and manages to reach the bottleneck bandwidth of 1 Mb/s in around 35 seconds for the case of linear increase and 45 seconds with the exponential increase.

Fig. 4.3(b) depicts the rate increase behavior when simulating an RTP sender altering its transmission behavior after receiving an RTCP feedback message from a receiver. The sender and receiver are connected through a link of 1 Mb/s bandwidth with a round trip delay of 100 msec. We notice a slight difference between the calculated and simulated behavior. This results from the fact that the interval between sending two RTCP messages is not exactly five seconds but might vary. However this has no great effect on the performance of the scheme. While Eqn. 4.5 and Eqn. 4.4 are ideal for the case of a single



(a) Calculated increase behavior



(b) Simulated increase behavior

Figure 4.3: Dynamical additive increase rate (linear and exponential case)

flow passing a bottleneck router, the actual challenge is estimating the fair bandwidth share when several flows compete for the available bandwidth simultaneously. While a single flow will not lead to an overflow of the bottleneck router, this is possible for the case of several flows increasing their transmission rate simultaneously. The highest overall increase in the load is achieved when all competing flows increase their rate simultaneously. In this case the additional load (L) introduced to the link is:

$$L = \sum_{i=1}^{n} r_{\mathrm{i}} \times \left(1 - \frac{r_{\mathrm{i}}}{R}\right) \tag{4.7}$$

for the linear case and

$$L = \sum_{i=1}^{n} r_{\mathrm{i}} \times \left(1 - \exp^{-(1-\frac{r_{\mathrm{i}}}{R})}\right) \tag{4.8}$$

for the exponential increase.

For the case of n flows and considering that the bandwidth is equally distributed each flow would be getting a share of $\frac{R}{n}$. Substituting this value in Eqn. 4.7 and Eqn. 4.8

$$L = n \times \frac{R}{n} \times (1 - \frac{\frac{R}{n}}{R}) \tag{4.9}$$

for the linear case with L having a maximum value for $(n \rightarrow \infty)$

$$\lim_{n \rightarrow \infty} L = R \tag{4.10}$$

and

$$L = n \times \frac{R}{n} \times (1 - \exp^{-(1 - \frac{\frac{R}{n}}{R})}) \tag{4.11}$$

for the exponential increase with L having a maximum value for $(n \rightarrow \infty)$

$$\lim_{n \rightarrow \infty} L = R \times (1 - \exp^{-1}) \tag{4.12}$$

Fig. 4.4 depicts the maximum possible overload value scaled by the maximum available rate (R) calculated for both the linear and exponential increase. We notice that while the exponential increase shown in Fig. 4.3 shows a longer transient phase it resulted here in a lower maximum overload and thus is preferable for avoiding congestion in the network.

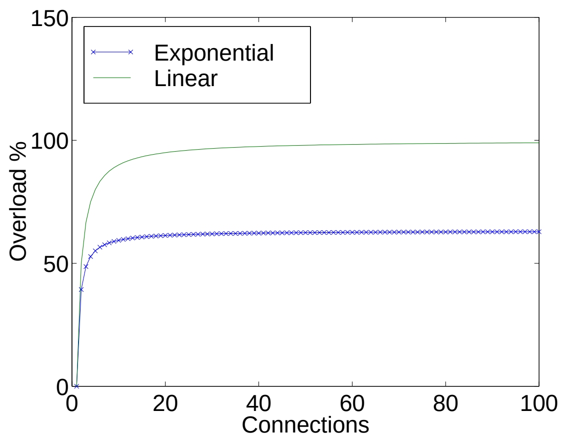


Figure 4.4: Overload caused by all senders increasing their rate simultaneously

Both linear and exponential increase approaches fulfill the requirements of fast increase during the transient phase and only minimal rate changes during the steady state. However, they fail to provide the third requirement of higher increase rates for flows with lower rates. That is, while the percentage of the rate increase is lower for flows with higher rates, the actual increase value might be larger. This might result in unfair bandwidth distribution for flows starting with a lower transmission rate. To avoid this situation we propose to use a constant initial increase value $\dot{R}_{\text{ai}}$ as the basis for the rate increase which is then increased inversely proportional to the current transmission rate. That is,

$$R_{\text{ai}_1} = \dot{R}_{\text{ai}} + (1 - \frac{r_0}{R}) \times \dot{R}_{\text{ai}} \tag{4.13}$$

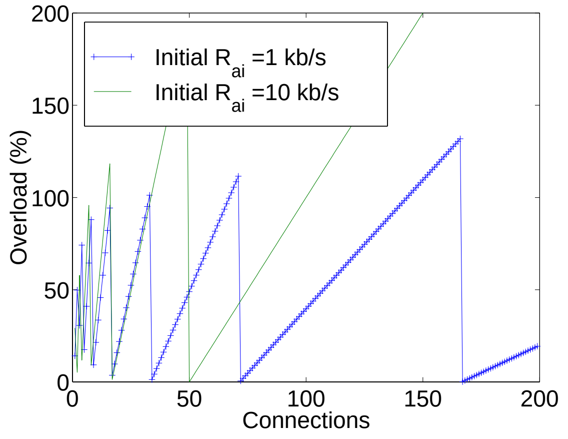
$$R_{\text{ai}_n} = R_{\text{ai}_{n-1}} + (1 - \frac{r_{\text{n-1}}}{R}) \times R_{\text{ai}_{n-1}} \tag{4.14}$$

and the rate (r) is increased as follows

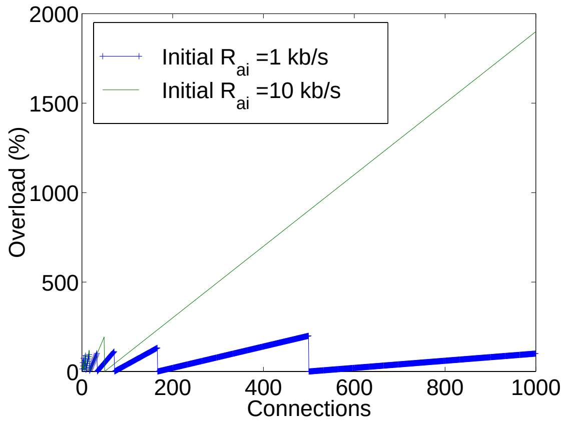
$$r_{\text{n}} = r_{\text{n-1}} + R_{\text{ai}_n} \tag{4.15}$$

With this approach, for the case of two flows having different transmission rates and starting the increase phase simultaneously, the flow with the lower staring rate would increase its rate faster. $\dot{R}_{\text{ai}}$ should be chosen small enough to prevent high oscillations. In this work, we will be looking at the effects of $\dot{R}_{\text{ai}}$ values in between 1 and 10 kb/s.

However, R_{ai} does not converge to 0 in this case which might result in high maximum overload situations. Fig. 4.5 depicts the overload calculated for the case of different numbers of senders increasing their transmission rate using Eqn. 4.13 and starting at the same time with an initial transmission rate of 1 b/s and an initial additive increase rate $(\dot{R}_{\text{ai}})$ of 1 kb/s and 10 kb/s. The overload is calculated as the difference between the first transmission rate (r_{i}) that is larger than the theoretically fair share $(\frac{R}{n})$ when distributing the available bandwidth among n flows and $\frac{R}{n}$. Figures 4.5(a) and 4.5(b) show that the maximum overload varies with the number of competing flows and the initial increase rate $(\dot{R}_{\text{ai}})$. For the case of a large number of senders the overload increases linearly with the number of flows.



(a) Overload caused by up to 200 flows



(b) Overload caused by up to 1000 flows

Figure 4.5: Overload caused by increasing the rate using Eqn. 4.13

To avoid the high overload problem calculated with Eqn. 4.13 but maintain its advantages in terms of fairness we combine it with Eqn. 4.5 to give an increase behavior as follows:

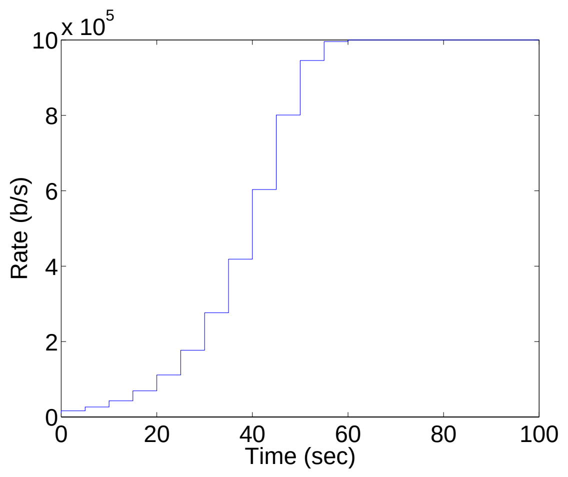
$$R_{\text{ai}_n} = \min(R_{\text{ai\_ add}_n}, R_{\text{ai\_ exp}_n}) \tag{4.16}$$

with

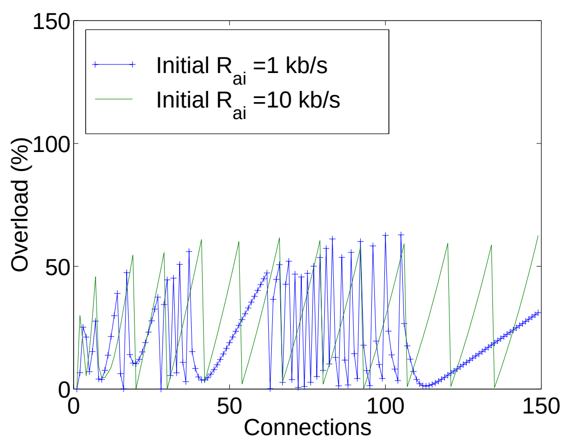
$$R_{\text{ai\_ add}_n} = (1 - \frac{r_{\text{n-1}}}{R}) \times R_{\text{ai}_{n-1}} \text{and } R_{\text{ai\_ exp}_n} = (1 - \exp^{-(1-\frac{r_{\text{n-1}}}{R})}) \times r_{\text{n-1}} \tag{4.17}$$

Combining both approaches results in an increase behavior that combines the positive convergence feature of the exponential increase in Eqn. 4.5 and the fairness of Eqn. 4.13.

Fig. 4.6(a) depicts the increase behavior calculated for a sender using Eqn. 4.16 for increasing its transmission behavior. The sender starts with a transmission rate of 10 kb/s and an initial additive increase rate of 10 kb/s. The sender manages to fully utilize the entire link of 1 Mb/s in around 50 seconds. We notice that the increase behavior is very similar to the increase behavior depicted in Fig. 4.3(a). The overload depicted in Fig. 4.6(b) shows now a maximum overload value of 63% which is equal to the overload calculated using Eqn. 4.5.



(a) Increase behavior using the combined approach



(b) Maximum overload using the combined approach

Figure 4.6: Overload caused by increasing the rate using Eqn. 4.16

To ensure an algorithm's fairness towards competing TCP connections, the RTP sender should not send more packets in between the reception of two RTCP messages than a TCP connection sharing the same link and having a similar round trip time would. So for an average period of T seconds in between the reception of two RTCP messages and a round trip delay of (τ) the TCP connection would increase its transmission window by P packets during a period of T seconds with no observed losses with P determined as

$$P = \sum_{i=0}^{T/\tau} i = \frac{(\frac{T}{\tau} + 1) \times \frac{T}{\tau}}{2} \tag{4.18}$$

with the window size being increased by one packet of size (M) each round trip delay. The round trip delay (τ) can be estimated by the sender using the RTCP timing information. The packet size (M) can be assumed to be known for the sender who generates those packets. Averaged over T, the RTP sender should maximally increase its transmission rate by

$$R_{\text{ai}} = M \times \frac{P}{T} \rightarrow M \times \frac{\frac{T}{\tau} + 1}{2 \times \tau} \tag{4.19}$$

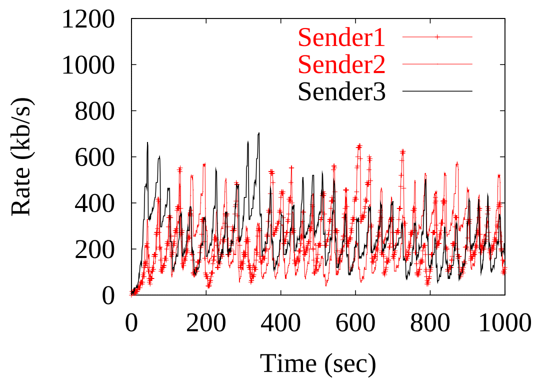
4.3.2.2 Options for Designing the Reduction Phase

While the choice of the increase behavior determines the ability of an adaptation scheme to benefit from the available resources it is the reduction behavior that governs the scheme's ability to avoid overload situations and network congestion.

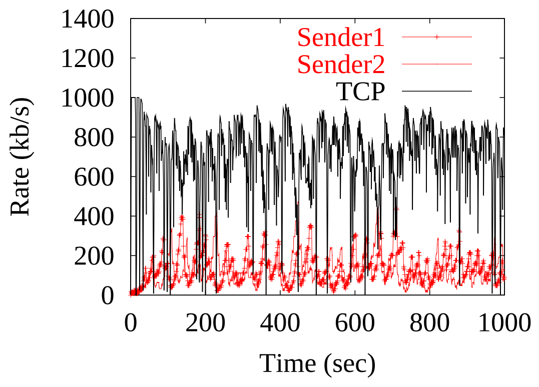
In this section, we present a few possible options for designing the reduction phase and introduce a novel TCP-friendly reduction approach. In this context, we mainly consider the fairness of the discussed approaches, their smoothness in the steady state and the network performance in terms of bandwidth utilization and losses.

Constant Multiplicative Decrease Factor

With this approach the transmission rate is reduced by multiplying it by a constant factor when congestion is indicated. The work done in [34] shows analytically, that using a constant multiplicative decrease factor results in equal bandwidth distribution among competing flows. As flows with a higher bandwidth share reduce their share faster during the decrease phase, the bandwidth shares of all flows will finally converge to the same value.



(a) 3 RTP flows



(b) 2 RTP and 1 TCP connection

Figure 4.7: Bandwidth distribution for adaptive flows with a multiplicative reduction factor of 0.5

Adaptation schemes using a constant reduction factor usually use either a multiplicative reduction factor of 0.875 [78, 148, 25] or 0.5 [19, 44]. 0.875 is often chosen because it can be represented as $(1 - 2^{-3})$ and the multiplication can thus be performed without floating point hardware by simple logical shift instructions. Algorithms using 0.5 as a reduction factor aim to be TCP-friendly by using the same reduction factor as TCP which halves its transmission window after discovering packet losses.

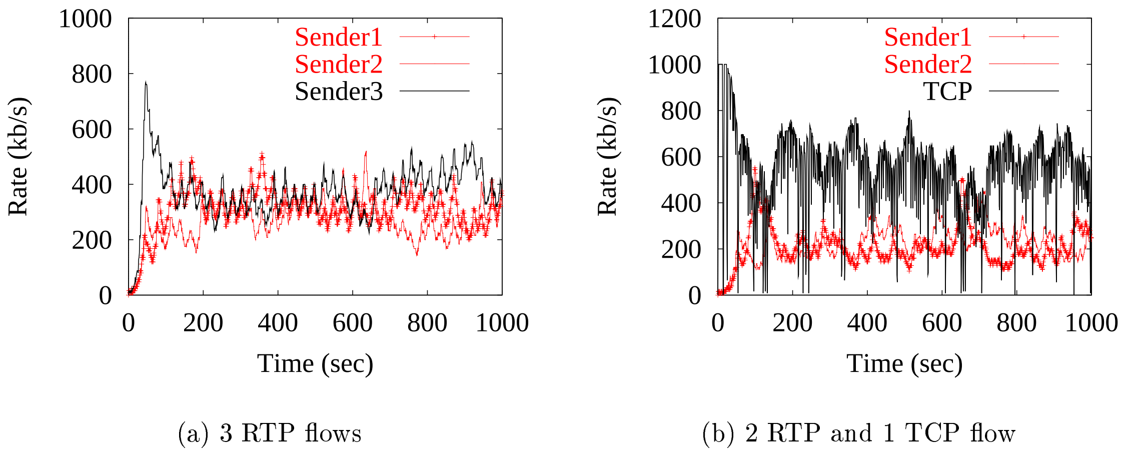
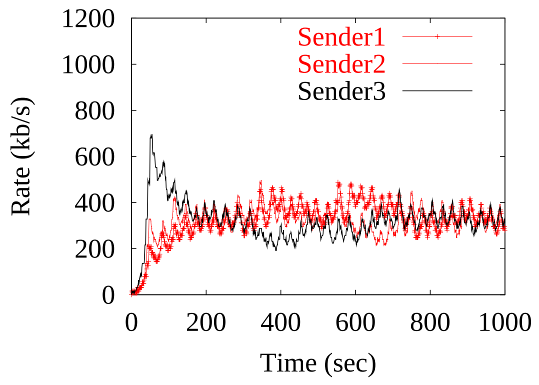


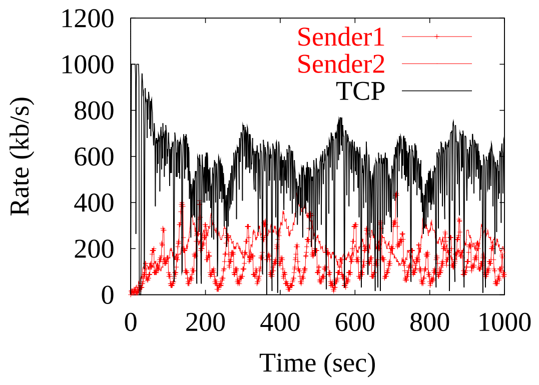
Figure 4.8: Bandwidth distribution for adaptive flows with a multiplicative reduction factor of 0.875

To test the effects of using constant multiplicative decrease factors we simulated the case of three RTP flows sharing a 1 Mb/s link with the router using RED for buffer management, see Fig. 2.5 with $(m = 3, n = k = 0)$. The adaptive RTP flows used a dynamical increase rate approach as was described by Eqn. 4.16 and a constant reduction factor (F_{mr}). Here, the RTP senders used the RTCP feedback mechanisms for determining the loss and delay situation in the network. Additionally, the bottleneck bandwidth (R) was assumed to be measurable and known for the senders. The round trip delay was set to 100 msec and the gateway was a RED router to ensure that all flows receive the same loss rate and avoid synchronization among them. Fig. 4.7 and Fig. 4.8 depict the adaptation behavior when using a reduction factor of 0.5 and 0.875. Note, that in contrast to [19, 25] we did not use a minimum loss threshold below which the sender does not need to consider losses. As TCP connections react to any losses by reducing their transmission window, non-TCP connections should do the same. Otherwise, the adaptation scheme can not be TCP-friendly as the simulations in Sec. 4.2 suggest. In both cases, see Fig. 4.7(a) and Fig. 4.8(a), the bandwidth is fairly distributed with each flow receiving around 30% of the available link bandwidth.

The major drawback of using constant reduction factors is the inability of the adaptation schemes to correctly consider the actual congestion level in the network when reducing the transmission rate. Thus, even for small losses the transmission rate is considerably reduced. For the case of TCP, the sender can rapidly recover from loss situations and start increasing its transmission rate after a time interval in the scale of a few round trip delays. For the case of RTCP-based adaptation the recovery process can only start after a period in the range of a few up to several seconds. This allows the TCP connection to grab a larger share of the bandwidth than the RTP flows as depicted in Fig. 4.7(b). Using larger reduction factors results in reduced oscillations of the adapting flows as they



(a) 3 RTP flows



(b) 2 RTP and 1 TCP flow

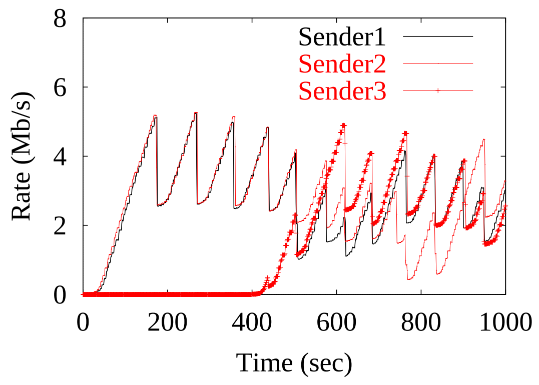
Figure 4.9: Bandwidth distribution for adaptive flows with a multiplicative reduction factor of 0.9

react less severely to the loss notifications. Further, better bandwidth distribution among the competing TCP and RTP flows is achieved, see Fig. 4.9. However, this also results in lower convergence between flows with a high bandwidth share and flows just starting to transmit data. In Fig. 4.10(b) a link is shared between two flows for 400 seconds. Afterwards a new flow starts transmitting data. However, for the case of a reduction factor of 0.9 the new flow can only increase its transmission rate very slowly as the other two flows release their bandwidth share at a very low rate.

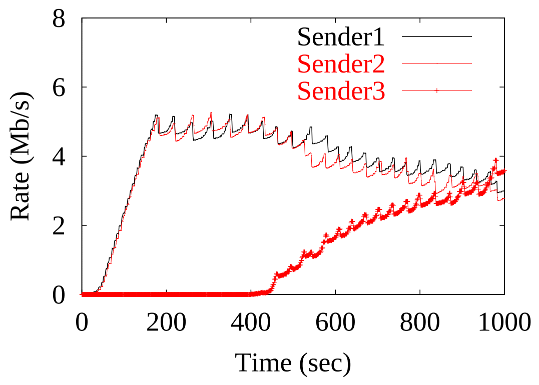
TCP-Based Reduction

To test the effects of using Eqn. 2.6 as the basis for rate reduction, as was suggested by [130] we simulated the configuration presented in Fig. 2.5 with 15 RTP senders competing with a TCP connection ($m = 15$ and $n = 1$). Based on the RTCP feedback the senders calculate the round trip delay and loss fraction of their transmitted data. For the case of no losses, the senders can increase their transmission rates using the dynamic increase approach based on Eqn. 4.16. For the case of losses, the senders estimate the round trip delay and timeout value similar to TCP, see Sec. 2.1.2.1. Using those values and by setting D to one which indicates that TCP acknowledges each packet, the senders can now estimate the TCP throughput under similar loss and delay conditions. The round trip delay is set initially to 500 ms and the timeout value (t_{out}) to 2 seconds.

Fig. 4.11 shows the behavior of two of the RTP flows over a simulation period of 1500 seconds. In tables 4.1 and 4.2 the average bandwidth and standard deviation for all the competing flows are presented. The RTP flows receive an equal bandwidth share with a



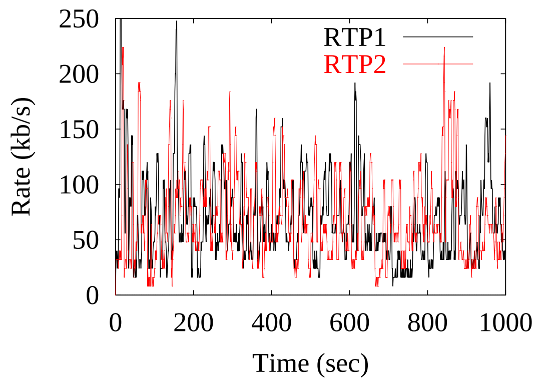
(a) Reduction factor= 0.5



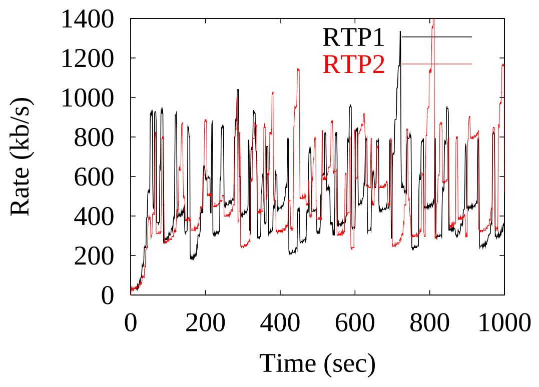
(b) Reduction factor= 0.9

Figure 4.10: Effects of the multiplicative decrease factor on fairness towards late starters

variance of around 10% and the transmission rate oscillates over a wide range of around 50% of the average value. The TCP connection receives a bandwidth share of around three times as high as the RTP flows.



(a) R=1 Mb/s



(b) R=10 Mb/s

Figure 4.11: Bandwidth distribution with a TCP-model based adaptation

The TCP connection uses the same transmission rate as the RTP flows for the same losses. However, due to its short adaptation intervals it manages to grab a larger bandwidth share during underload situations. Thereby, the average rate of the TCP connection is much higher than that of an RTP flow. Additionally, the rather oscillative behavior indicated by the large standard deviation values is inappropriate for multimedia communication. Taking the example of video communication the oscillative transmission rate would result in a rapidly changing frame rate at the receiver which is in general rather

Flow	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	TCP
Mean (b/s)	66000	59888	72528	58733	61765	58376	63389	69470	67013	61536	62441	65802	59269	65936	66885	$\boxed{\mathbf{139720}}$
Deviation (b/s)	38872	37628	40676	38430	36470	33258	37954	450968	41548	39610	36367	9393	36981	35400	40607	133670

Table 4.1: Bandwidth distribution and standard deviation for TCP-based adaptation with ($R = 1$ Mb/s)

Flow	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	TCP
Mean (b/s)	545130	44806	493500	523824	515720	520752	510750	517120	494550	536440	541590	462592	577820	531120	511250	$\boxed{\mathbf{1927600}}$
Deviation (b/s)	267260	211000	228970	265750	232050	254640	234700	214330	230770	249570	254210	221190	274080	232350	24680	1672000

Table 4.2: Bandwidth distribution and standard deviation for TCP-based adaptation with ($R = 10$ Mb/s)

annoying.

TCP-Similar Reduction Approach

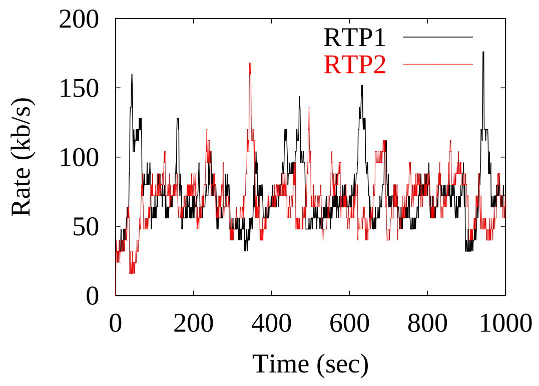
On the one hand, using a constant reduction factor does not take the actual congestion level into account and results thus in a bandwidth share for the RTP flows lower than a TCP connection would get. On the other hand, using the TCP-based reduction approach does not take the characteristics of the real time stream into account and results in a highly oscillative behavior. As a combination between the two approaches we present in this section another approach with which the sender reduces its rate multiplicatively but still takes the TCP model into account.

From Eqn. 2.6 and Eqn. 2.5 it is evident that the overall TCP throughput is inversely proportional to the square root of the loss values. Hence, we propose to reduce the rate of the RTP flows in a similar manner to TCP instead of using the exact model. Thus, after receiving a loss notification from the receiver, the sender can reduce its transmission rate as follows:

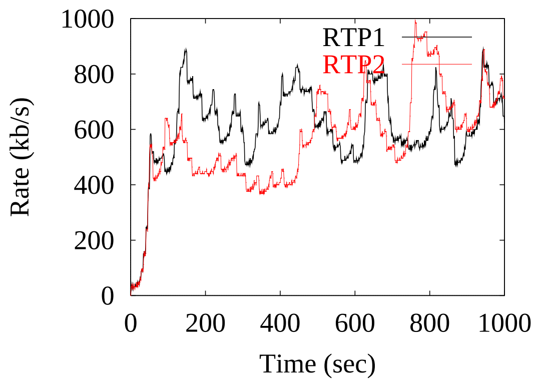
$$r_n = r_{n-1} \times (1 - \sqrt{l}) \tag{4.20}$$

Taking Eqn. 2.6 into account the sender needs only to reduce its transmission rate during loss phases if its current transmission rate is higher than the TCP-rate determined by Eqn. 2.6.

To test this reduction behavior we repeat the simulations of Sec. 4.3.2.2 with 15 RTP senders sharing a bottleneck link with a TCP connection. The results depicted in Fig. 4.12 and Tab. 4.3 and 4.4 reveal that while the TCP flow still receives a higher bandwidth share than the RTP flows, the difference between the share of the TCP connection and an RTP flow is much lower than in Tab. 4.1 and 4.1. Of even more significance is the considerably reduced variance of the achieved transmission rate of the RTP senders which is reduced by more than 100% compared to the results presented in Tab. 4.1 and 4.1. Both effects of better fairness and lower variance result from the slower reduction behavior which partially compensates for the slower increase behavior of the RTP flows.



(a) R= 1 Mb/s



(b) R= 10 Mb/s

Figure 4.12: Bandwidth distribution with the TCP-similar approach

Flow	1	2	3	4	5	6	7	8
Mean (b/s)	66710	67360	65595	64064	66448	65909	63296	63669
Deviation (b/s)	21322	16844	20815	18084	19149	19336	16328	19723
Flow	9	10	11	12	13	14	15	TCP
Mean (b/s)	64645	66080	61355	65664	66885	64549	71952	$\boxed{\mathbf{75345}}$
Deviation (b/s)	20064	23145	21822	19316	18164	1941	24204	65644

Table 4.3: Bandwidth distribution and standard deviation for TCP-similar adaptation with ($R = 1$ Mb/s)

4.3.3 Rate Adjustment with LDA

To conclude the results of the rate adaptation behavior investigated in Sec. 4.3.2 this section summarizes the rate adaptation with LDA.

The sender starts transmitting data with an initial transmission rate of $\dot{R}$ and an initial additive increase rate ($\dot{R}_{\text{ai}}$). After receiving an RTCP message from the receiver indicating a no loss situation the sender calculates an additive increase rate (R_{ai_n}) as was described in Sec. 4.3.2.1 as

$$R_{\text{ai}_n} = \min(R_{\text{ai\_add}_n}, R_{\text{ai\_exp}_n} R_{\text{ai\_TCP}_n}) \qquad (4.21)$$

Flow	1	2	3	4	5	6	7	8
Mean (b/s)	57034	58857	579310	539376	587140	576870	581312	586620
Deviation (b/s)	1.3016	1.3377	1.5655	1.2820	1.3904	1.4984	1.4251	1.4388
Flow	9	10	11	12	13	14	15	TCP
Mean (b/s)	580800	612460	586500	571936	572416	554820	621860	**1189300**
Deviation (b/s)	1.5868	1.4427	1.4088	1.3957	1.4713	1.4646	1.4678	1.1797

Table 4.4: Bandwidth distribution and standard deviation for TCP-similar adaptation with ($R = 10$ Mb/s)

with

$$R_{\text{ai\_add}_n} = (2 - \frac{r_{\text{n-1}}}{R}) \times R_{\text{ai}_{n-1}} \tag{4.22}$$

$$R_{\text{ai\_exp}_n} = (1 - \exp^{-(1-\frac{r_{\text{n-1}}}{R})}) \times r_{\text{n-1}} \tag{4.23}$$

$$R_{\text{ai\_TCP}_n} = M \times \frac{\frac{T}{\tau} + 1}{2} \tag{4.24}$$

with $r_{\text{n-1}}$ as the current transmission rate, $R_{\text{ai}_{n-1}}$ as the current additive increase rate, R as the bottleneck bandwidth, M as the packet size, T as the length of the adaptation interval and τ as the estimated round trip delay.

If losses (l) were indicated in the RTCP messages with ($r_{\text{i-1}} > r_{\text{TCP}}$) then the transmission rate is reduced to:

$$r_{\text{n}} = \max(r_{\text{n-1}} \times (1 - \sqrt{l}), r_{\text{TCP}}) \tag{4.25}$$

with r_{TCP} as the rate calculated using Eqn. 2.6. R_{ai_i} is reset to $\dot{R}_{\text{ai}}$. For the case of ($r_{\text{n-1}} < r_{\text{TCP}}$) the sender calculates a new R_{ai_n} as described in Eqn. 4.21 and increase its transmission rate to

$$r_{\text{n}} = \min(r_{\text{n-1}} + R_{\text{ai}_n}, r_{\text{TCP}}) \tag{4.26}$$

4.4 Performance of the Loss-Delay Based Adjustment Algorithm

When designing an adaptive control scheme, following goals need to be considered:

- the scheme should result in a low packet loss ratio,
- achieve high overall bandwidth utilization,
- fairly distribute bandwidth between competing flows,
- and scale for large multicast groups.

In this section, we investigate the performance of the LDA algorithm with regard to those different requirements using both simulations as well as measurements on a real network. Scalability issues will be discussed in chapter 6 when considering the aspect of adaptation for multicast communication.

For testing the performance of LDA we used the dumb-bell topology described in Sec. 2.3.1.

In Sec. 4.4.1, we first look at the performance of LDA in terms of fairness of LDA flows among each other and the effects of the initial adaptation parameters such as the initial additive increase rate ($\dot{R}_{\text{ai}}$) and the initial transmission rate ($\dot{R}$). Additionally, we investigate the effects of errors in the bottleneck bandwidth estimation and differences in the round trip propagation delays on the fairness of LDA. In Sec. 4.4.2 the TCP-friendliness of LDA under different simulation parameters and buffer management schemes is investigated. To test the performance and TCP-friendliness of LDA when deployed over the Internet, the results of a wide range of measurements are presented in Sec. 4.5. Finally, in Sec. 4.6, the performance of LDA is compared to a number of recently proposed schemes for TCP-friendly congestion control.

4.4.1 Fairness and Convergence of the LDA Algorithm

In this section, we tested the inter-protocol performance of LDA. That is, the fairness and performance of competing LDA streams. As a test topology we used the configuration depicted in Fig. 2.5 with ($k = n = 0$) and m senders using LDA to control the transmission rate of unicast data flows. We tested the performance of LDA in terms of fairness and link bandwidth utilization under different conditions of varying round trip delays, different link bandwidths and adaptation parameters such as initial transmission rate of the senders and the initial additive increase rate.

To make sure that all streams are treated similarly in terms of drop probability and avoid the effects of synchronization that might arise when using drop tail buffer management at the bottlenecked router we used a random early drop based router, see Sec. 2.1.1.1. The effects of using a drop tail router are presented in Sec. 4.4.2.2.

4.4.1.1 General Performance Tests of LDA

In this part, we tested the performance of LDA in terms of mean sending rate of each flow (r) in kb/s, the mean loss value (l) and the standard deviation ($\overline{\sigma}$) of the average rate of the single flows from the value (r) obtained by averaging over the average rates of all flows. The performance of LDA was investigated with varying round trip propagation delays (τ), buffering delays (τ_q), bottleneck bandwidths (R) and number of RTP flows (m). All flows had the same values for the round trip propagation delay (τ), initial additive increase rate ($\dot{R}_{\text{ai}}$) of 5 kb/s and an initial transmission rate ($\dot{R}$) of 10 kb/s.

The results presented in Tab. 4.5 up to 4.12 suggest the efficiency of LDA in achieving a high network utilization level and a mean transmission rate close to the theoretically calculated one ($\frac{R}{m}$). The deviation values ($\overline{\sigma}$) of less than 10% from the overall average rate (r) indicates that the bandwidth is distributed in a similar way among the competing flows. We notice that for both increasing the round trip propagation delay as well as

R (kb/s)	10000.00				1500.00			
τ (sec)	r (kb/s)	$\overline{\sigma}$ (kb/s)	l	u	r (kb/s)	$\overline{\sigma}$ (kb/s)	l	u
0.1	330.62	21.91	0.082	0.79	55.58	4.01	0.179	0.80
0.4	332.09	22.43	0.052	0.84	52.84	3.93	0.170	0.77
1.0	333.18	18.81	0.037	0.86	50.19	3.97	0.170	0.74

Table 4.5: Performance of LDA with 27 LDA competing flows and τ_q set to 0.5

R (kb/s)	10000.00				1500.00			
τ (sec)	r (kb/s)	$\overline{\sigma}$ (kb/s)	l	u	r (kb/s)	$\overline{\sigma}$ (kb/s)	l	u
0.1	118.82	9.04	0.115	0.84	20.49	3.03	0.277	0.81
0.4	114.23	9.59	0.107	0.81	19.81	3.01	0.264	0.79
1.0	111.10	6.82	0.088	0.80	18.65	2.92	0.272	0.74

Table 4.6: Performance of LDA with 54 LDA competing flows and τ_q set to 0.5

R (kb/s)	10000.00				1500.00			
τ (sec)	r (kb/s)	$\overline{\sigma}$ (kb/s)	l	u	r (kb/s)	$\overline{\sigma}$ (kb/s)	l	u
0.1	119.04	8.46	0.117	0.83	20.69	2.99	0.268	0.82
0.4	114.07	8.99	0.104	0.81	19.75	3.13	0.265	0.78
1.0	111.56	7.19	0.087	0.81	18.28	3.23	0.281	0.72

Table 4.7: Performance of LDA with 81 LDA competing flows and τ_q set to 0.5

R (kb/s)	10000.00				1500.00			
τ (sec)	r (kb/s)	$\overline{\sigma}$ (kb/s)	l	u	r (kb/s)	$\overline{\sigma}$ (kb/s)	l	u
0.1	118.50	8.91	0.120	0.83	20.24	3.53	0.283	0.79
0.4	114.10	9.08	0.101	0.82	19.44	3.14	0.275	0.77
1.0	112.11	7.93	0.086	0.81	17.87	2.74	0.283	0.70

Table 4.8: Performance of LDA with 108 LDA competing flows and τ_q set to 0.5

R (kb/s)	10000.00				1500.00			
τ (sec)	r (kb/s)	$\overline{\sigma}$ (kb/s)	l	u	r (kb/s)	$\overline{\sigma}$ (kb/s)	l	u
0.1	354.79	26.34	0.035	0.91	59.90	3.95	0.156	0.90
0.4	346.31	27.73	0.026	0.88	56.24	2.83	0.096	0.91
1.0	346.73	26.11	0.021	0.90	52.51	3.93	0.095	0.85

Table 4.9: Performance of LDA with 27 LDA competing flows and τ_q set to 0.1

R (kb/s)	10000.00				1500.00			
τ (sec)	r (kb/s)	$\overline{\sigma}$ (kb/s)	l	u	r (kb/s)	$\overline{\sigma}$ (kb/s)	l	u
0.1	131.77	4.68	0.070	0.99	24.89	1.81	0.275	0.98
0.4	117.96	8.02	0.056	0.89	20.94	1.98	0.228	0.88
1.0	115.45	6.91	0.050	0.88	18.79	2.23	0.221	0.80

Table 4.10: Performance of LDA with 54 LDA competing flows and τ_q set to 0.1

R (kb/s)	10000.00				1500.00			
τ (sec)	r (kb/s)	$\overline{\sigma}$ (kb/s)	l	u	r (kb/s)	$\overline{\sigma}$ (kb/s)	l	u
0.1	133.34	4.11	0.075	0.99	25.05	1.99	0.282	0.98
0.4	118.56	6.82	0.053	0.89	20.75	2.07	0.232	0.87
1.0	115.45	8.80	0.051	0.88	19.00	1.98	0.213	0.82

Table 4.11: Performance of LDA with 81 LDA competing flows and τ_q set to 0.1

R (kb/s)	10000.00				1500.00			
τ (sec)	r (kb/s)	$\overline{\sigma}$ (kb/s)	l	u	r (kb/s)	$\overline{\sigma}$ (kb/s)	l	u
0.1	132.05	4.50	0.071	0.99	24.93	2.10	0.279	0.98
0.4	118.98	7.05	0.050	0.90	21.06	2.00	0.231	0.89
1.0	115.53	7.53	0.051	0.88	18.90	2.41	0.215	0.81

Table 4.12: Performance of LDA with 108 LDA competing flows and τ_q set to 0.1

increasing the maximum buffering delay, the average transmission rate decreases as would be expected from a TCP connection, due to the longer recovery period. For the case of a high number of flows ($m = 81$) and a small bottleneck bandwidth ($R = 1.5$ Mb/s) the senders actually use a transmission rate higher than the theoretical value of 18.5 kb/s which results in high loss values of around 20%. This, however, resulted in part from the chosen packet size of 8000 bits and the initial increase rate of 5 kb/s. Thus, losing one packet at a rate of 3 packets/s would result in a loss value of 33%. Additionally, using an R_{ai} with an initial value of 5 kb/s in this case leads to relatively large changes in the transmission rate and hence to high losses and an average transmission rate higher than the theoretically appropriate one.

4.4.1.2 Effects of the Accuracy of the Bottleneck Bandwidth Estimation on the Performance of LDA

The primary goal of the bottleneck measurement with LDA is to make sure that the sender does not transmit data at a rate higher than the capacity of the smallest link along the path to the receiver. Additionally, the bottleneck rate is used for determining the additive increase rate. Due to the usage of different measurement and filtering approaches, flows competing for the same resources might estimate different values for the bottleneck bandwidth. Hence, in this case the competing flows would use different increase factors.

To test the effects of inaccurate bottleneck estimations on the inter-flow fairness of LDA, we used the simulation topology of Fig. 2.5 with R set to 1 Mb/s, τ set to 0.4 seconds, τ_q set to 0.1 seconds and m set to three. Each flow used a different value of the bottleneck rate estimation (B). Fig. 4.13 depicts the transmission rates of the three flows. We can observe that the bandwidth share of the different flows is affected by their estimation of the bottleneck bandwidth. Hence, the flow with an estimation of twice the actual bandwidth receives the highest share. Note however, that the differences are not directly proportional to the estimation. That is, the share of the flow with an estimation of ($B = 2$ Mb/s) does not receive four times as much as the flow with ($B = 0.5$

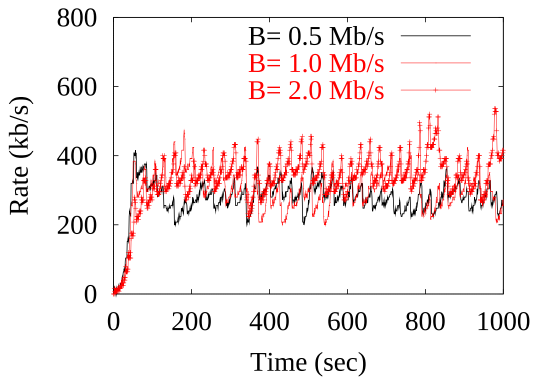


Figure 4.13: Effects of bottleneck estimations on the inter-flow fairness of LDA

Mb/s). Testing for small variations in the bandwidth estimation did not show a significant difference between the competing flows. Actually for higher numbers of competing flows the effect shown in Fig. 4.13 was further reduced. So, while it is important to get an accurate estimation of the bottleneck in order to avoid overloading the smallest traversed connection, small inaccuracies in the estimation do not have significant effects on the performance of LDA.

4.4.1.3 Effects of the RTCP Intervals on the Performance of LDA

With most of the adaptation schemes presented in Sec. 3.1.2 the senders adapt their transmission behavior based on feedback from the receivers sent in short intervals in the range of one or a few round trip delays. This is particularly important for the case of reliable transport where the sender needs to retransmit lost packets. Additionally, with frequent feedback messages the sender can obtain up-to-date information about the round trip delay and hence use an increase in the round trip delay as an early indication of possible congestion.

With RTP, the interval between sending two RTCP messages is usually around five seconds. To investigate the possible benefits of using smaller intervals we simulated the topology depicted in Fig. 2.5 with 27 LDA flows competing for a bottleneck of 10 Mb/s. As background traffic we used 27 WWW servers. The round trip propagation delay τ was set to 0.4 seconds and the maximum queuing delay (τ_q) to 0.1 seconds. Each simulation was run for 500 seconds and the presented results are the average values of 10 simulation runs with different seed values.

Congestion in packet switched networks is manifested by buffer overflows at the routers. That is, a constantly filled buffer is a good indication of a highly congested network. Network underutilization is on the other hand indicated by empty buffers. To investigate the effects of changing the RTCP interval on the network congestion state, in our simulations we measured the time the buffer was observed to be occupied to a certain percentage of the maximum buffer, see Fig. 4.14. That is, for the case of RED buffer management, the results depicted in Fig. 4.14 reveal that 0 to 0.1 of the buffer was occupied for 48% of the simulation time for the case of an RTCP sending interval of 5 seconds and the range between 0.1 and 0.2 of the buffer was occupied 17% of the time. Fig. 4.14 suggests that for both cases of FIFO and RED buffer management, using larger RTCP intervals results in reduced network congestion as indicated by the lower occupancy times of the higher ranges of the buffer. So for the case of an RTCP interval of one second, the buffer was 2.4% of the simulation time above 90% occupied, whereas for intervals of five seconds this value is reduced to 1.8%. It is also interesting to see that with RED the buffer occupancy in the high region of above 80% is much lower than that of FIFO, indicating a more balanced load situation.

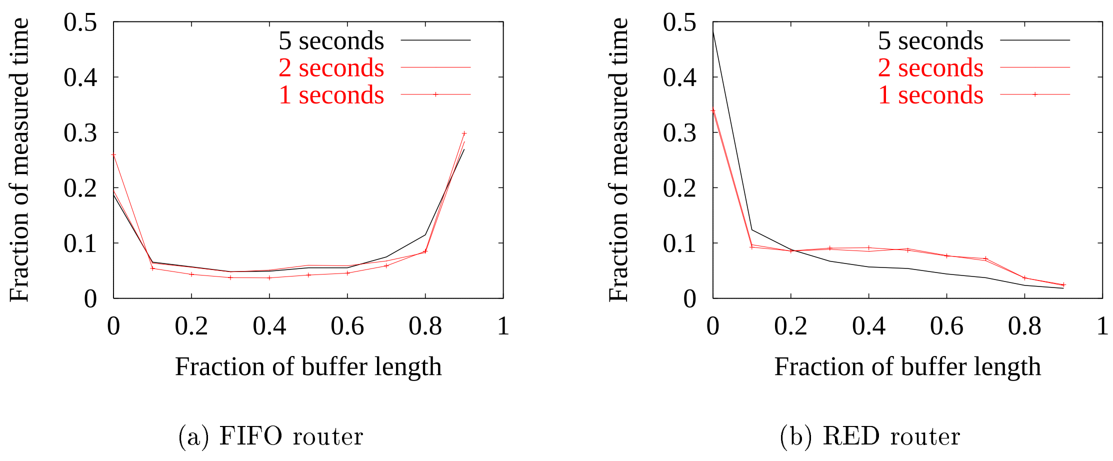


Figure 4.14: Effects of the RTCP intervals on the buffer occupancy

Tab. 4.13 presents the average bandwidth share of the flows (r) as well as the average utilization of the router (u) for both cases of using FIFO and RED for buffer management. The rather equal utilization and rate results suggest that reducing the RTCP interval does not improve the performance of LDA significantly. Actually, the reduced buffer occupancy values indicate that with larger RTCP values LDA is more conservative in its adaptation behavior and hence less affected by network instabilities and traffic burstiness.

	FIFO		RED	
Interval (sec)	r	u	r	u
5	126.8	0.991	105.0	0.971
2	122.4	0.993	118.1	0.991
1	121.2	0.997	107.2	0.997

Table 4.13: Performance of LDA for different RTCP intervals

4.4.1.4 Effects of the Initial Additive Increase Rate ($\dot{R}_{\mathrm{ai}}$) Value on the Performance of LDA

In contrast to various other schemes such as [25, 19], LDA dynamically determines the increase rate to use during underload situations. However, an initial value ($\dot{R}_{\mathrm{ai}}$) still needs to be defined as the starting point.

For testing the effects of the choice of $\dot{R}_{\text{ai}}$ on the performance of LDA we used the simulation topology of Fig. 2.5 with R set to 10 Mb/s, τ set to 0.4 seconds, τ_q set to 0.1 seconds and m set to 27 and 81. In each case a third of the flows used a different value of $\dot{R}_{\text{ai}}$. Fig. 4.15 depicts the transmission rates of three arbitrarily chosen flows each using a different $\dot{R}_{\text{ai}}$.

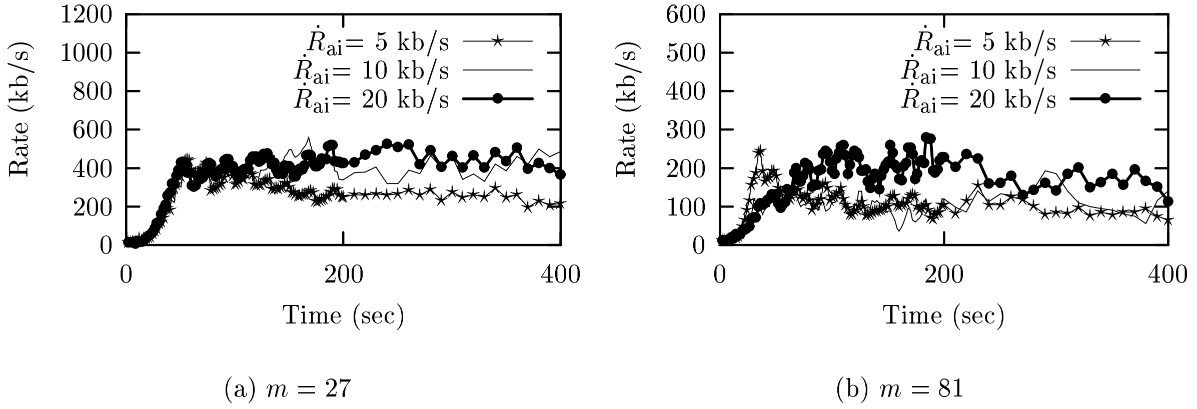


Figure 4.15: Effects of $\dot{R}_{\text{ai}}$ on the bandwidth distribution achieved with LDA

The results presented in Fig. 4.15(a) show that for the case of ($m = 27$), flows with a larger value of $\dot{R}_{\text{ai}}$ receive a higher bandwidth share. However, the larger $\dot{R}_{\text{ai}}$ also results in larger oscillations.

In the rest of the study, we will be using a $\dot{R}_{\text{ai}}$ of 5 kb/s which resulted in an acceptable bandwidth share, lower oscillations and was still small enough to accommodate the case of highly loaded networks.

4.4.1.5 Effects of the Initial Transmission Rate ($\dot{R}$) on the Performance of LDA

To test the effects of the initial transmission rate the senders use when starting to send data, we used the simulation topology of Fig. 2.5 with R set to 10 Mb/s, τ set to 0.4 seconds, τ_q set to 0.1 seconds and m set to 27 and 81. In each case a third of the flows used a different value of $\dot{R}$. Fig. 4.16 depicts the transmission rates of three arbitrarily chosen flows each using a different $\dot{R}$.

The presented results in Fig. 4.16 show that using a large $\dot{R}$ only results in a short transitional period of unfair bandwidth distribution. For the case of ($m = 27$), see Fig. 4.16(a), the different flows converge to similar shares after a period of 30 seconds. For the case of ($m = 81$), see Fig. 4.16(b), the transitional period is even reduced to less than 10 seconds which is about the time needed for the sender to receive an RTCP message with loss indication from the receiver.

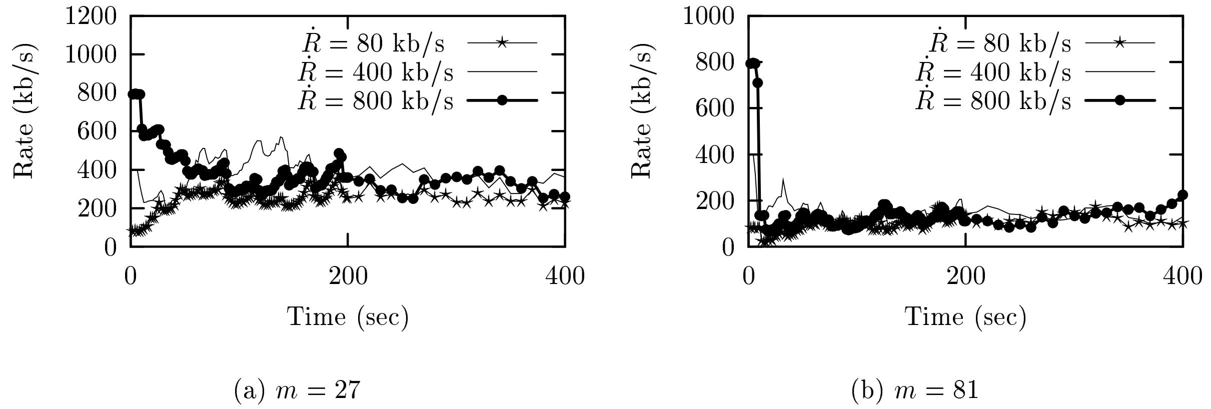


Figure 4.16: Effects of $\dot{R}$ on the bandwidth distribution achieved with LDA

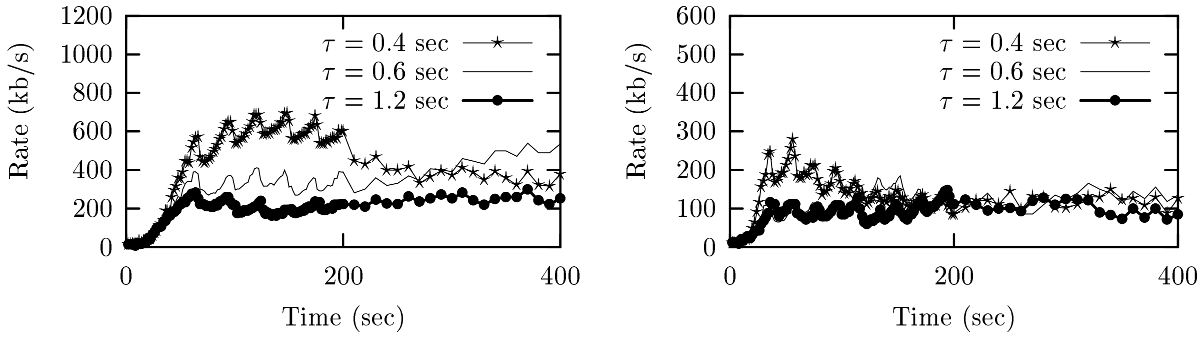
4.4.1.6 Effects of the Round Trip Propagation Delay on the Performance of LDA

Various simulative [171, 51] as well as analytical [110] studies show TCP's performance to depend on the round trip delays. A TCP sender increases its transmission window based on the acknowledgment packets from the receiver. Receiving an acknowledgment takes about one round trip delay. Thus, for longer round trip delays the sender increases its transmission window and hence its bandwidth share more slowly than a flow with a smaller round trip delay. The same applies for receiving loss notifications and recovery from losses.

To investigate the case of similar competing LDA flows with similar adaptation parameters of ($\dot{R}$ = 10 kb/s) and ($\dot{R}_{\text{ai}}$ = 5 kb/s) but with different round trip delays, we used the simulation topology of Fig. 2.5 with R set to 10 Mb/s, τ_q set to 0.1 seconds and m set to 27 and 81. In each case a third of the flows used a different value of τ. Fig. 4.17 depicts the transmission rates of three arbitrarily chosen flows each having a different τ. For the case of the less congested network with ($m = 27$), see 4.17(a), the adaptation behavior of LDA is influenced more by the lower limit determined by Eqn. 2.6 for the TCP rate under similar loss and delay values. With more competing flows with ($m = 81$) the limit determined by Eqn. 2.6 is more conservative than the transmission rate calculated by LDA. Thus, in this case the round trip delay plays a less important role in determining the transmission rates and in contrast to the results observed in Fig. 4.17(a) LDA achieves an equal distribution, see 4.17(b).

4.4.1.7 Performance of LDA for the Case of Late Joiners

An important evaluation criteria of any adaptation scheme is not just its efficiency in utilizing freely available network resources, but also its reaction to changes in the amount of available resources.



(a) $N = 27$ (b) $N = 81$

Figure 4.17: Effects of the propagation delay (τ) on the bandwidth distribution achieved with LDA

Using the simulation topology of Fig. 2.5 with R set to 10 Mb/s, τ_q set to 0.1 seconds, τ set to 0.4 seconds and all senders using the adaptation parameters of ($\dot{R} = 10$ kb/s) and ($\dot{R}_{\text{ai}} = 5$ kb/s) we simulated the case of a network with a changing number of competing flows. Fig. 4.18 shows the transmission rates of three senders starting at different time points over the entire simulation time of 1000 seconds. At time ($T = 0$ seconds) the first nine senders start transmitting data and manage to fully utilize the available 10 Mb/s in less than 100 seconds. At ($T = 200$ seconds) additional nine senders start transmitting data. At that time the first nine senders start decreasing their transmission rate until all competing flows receive an equal share. The same behavior can be observed at time point ($T = 500$ seconds), when the last nine senders start transmitting data.

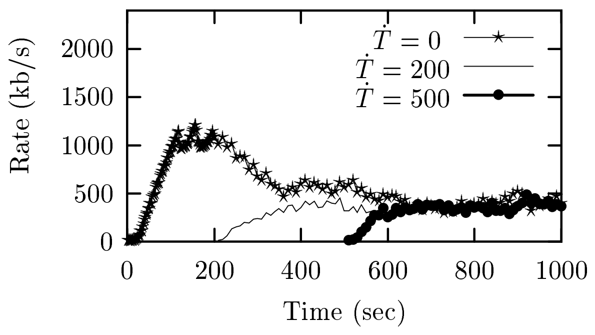


Figure 4.18: Performance of LDA in the case of late joiners

We notice that the start of new flows results in a transient period in which the bandwidth is unequally distributed. With LDA this transient period in the simulated scenario is less than 100 seconds long which allows for smooth reduction of the transmission rate of the flows at the high rate but still allows the new flows to rapidly increase their transmission rates.

4.4.2 Investigation of the TCP-Friendliness of LDA

To test the TCP-friendliness of LDA we used the topology depicted in Fig. 2.5 with different values for the round trip propagation delay (τ), buffering delay (τ_q), bottleneck bandwidth (R) and number of competing flows ($n = m = N$). Additionally, we simulated the cases of the bottleneck router using FIFO as well as RED for the buffer management.

4.4.2.1 Competing LDA and TCP Flows with a RED Router

Using RED for the buffer management prevents synchronization of the competing flows and ensures more evenly distributed losses. Similar to the simulations in Sec. 4.4.1 the minimum buffer threshold was set to 30% of the available buffer and the maximum threshold was set to 80%. The queuing weight (w_q) was set to 0.002 and the maximum drop probability (P_a) was set to 0.02. The bottleneck bandwidth (R) was set to 10 Mb/s.

Tables 4.14 up to 4.17 depict the results of the simulations in terms of achieved utilization (u), friendliness (F), average standard deviation ($\overline{\sigma}$) of the average bandwidth shares of the LDA flows from the overall average rate value as well as the average loss (l) and average transmission rate (r) of all the LDA flows. The presented results show that under all configurations a high bandwidth utilization level of more than 90% is achieved and the average loss is between 0.3% and 8%.

τ_q (sec)	0.10					0.50				
τ (sec)	r	$\overline{\sigma}$	F	l	u	r	$\overline{\sigma}$	F	l	u
0.1	469.12	18.95	0.72	0.007	0.99	616.36	146.52	1.22	0.004	0.99
0.2	500.98	74.76	0.87	0.005	0.95	694.00	176.17	1.62	0.004	0.99
0.4	711.76	108.00	1.97	0.005	0.95	790.42	169.83	2.41	0.004	0.99
1.0	871.57	84.29	4.26	0.005	0.94	893.38	62.48	4.46	0.005	0.96

Table 4.14: Achieved average rate and fairness for LDA with 9 TCP and 9 LDA competing flows and R set to 10 Mb/s

τ_q (sec)	0.10					0.50				
τ (sec)	r	$\overline{\sigma}$	F	l	u	r	$\overline{\sigma}$	F	l	u
0.1	170.79	4.72	0.81	0.039	0.98	170.29	10.45	0.82	0.016	1.00
0.2	149.26	5.16	0.70	0.029	0.94	173.79	15.36	0.86	0.012	0.99
0.4	161.37	11.00	0.84	0.016	0.94	193.61	18.39	1.07	0.010	0.99
1.0	236.40	23.12	2.02	0.011	0.93	250.23	22.19	2.17	0.010	0.97

Table 4.15: Achieved average rate and fairness for LDA with 27 TCP and 27 LDA competing flows and R set to 10 Mb/s

τ_{q} (sec)	0.10					0.50				
τ (sec)	r	$\overline{\sigma}$	F	l	u	r	$\overline{\sigma}$	F	l	u
0.1	101.71	3.26	1.05	0.083	0.98	90.93	3.78	0.89	0.040	1.00
0.2	87.17	3.25	0.83	0.061	0.97	90.02	5.67	0.88	0.036	0.99
0.4	81.30	3.68	0.78	0.039	0.96	92.70	6.12	0.96	0.026	0.99
1.0	107.63	8.78	1.49	0.024	0.94	115.62	9.37	1.70	0.027	0.95

Table 4.16: Achieved average rate and fairness for LDA with 54 TCP and 54 LDA competing flows and R set to 10 Mb/s

τ_{q} (sec)	0.10					0.50				
τ (sec)	r	$\overline{\sigma}$	F	l	u	r	$\overline{\sigma}$	F	l	u
0.1	75.43	2.59	1.22	0.116	0.98	65.73	2.37	0.99	0.071	0.99
0.2	64.43	2.32	0.94	0.091	0.98	63.00	3.15	0.94	0.056	0.99
0.4	57.91	2.46	0.82	0.063	0.97	62.94	3.86	0.97	0.043	0.99
1.0	69.19	5.35	1.29	0.036	0.95	72.71	5.59	1.42	0.044	0.95

Table 4.17: Achieved average rate and fairness for LDA with 81 TCP and 81 LDA competing flows and R set to 10 Mb/s

The friendliness factor (F) is determined as the ratio of the average LDA goodput to the average TCP goodput as was described in Eqn. 3.4. F varies depending on the chosen simulation parameters between 0.7 and 4.46. In general, for most of the simulated configurations, LDA achieves a friendliness factor of ($0.8 < F < 1.2$) which is rather close to the optimal value of ($F = 1$). We can notice, however, that in general F increases with increased round trip delay. This is especially obvious for the case of a large buffering delay (τ_{q}) of 0.5 seconds and a large propagation delay of ($\tau = 1$ second). In this case, F may get up to 4.5, indicating that the LDA controlled senders transmit at a rate more than four times higher than the competing TCP connections. Looking in more detail at simulation traces for the case of large round trip delays ($\tau = 1$ second) we discovered different aspects that contributed to this unfairness:

- The first problem stems from the design of the RTCP protocol. For example consider the configuration with N set to 9, τ set to one second, τ_q set to 0.1 seconds and R set to 10 Mb/s, in this case with each flow receiving an equal share the fair bandwidth share would be 556 kb/s. Using Eqn. 2.6 with D set to one, t_{out} set to one second and t_{RTT} set to one second the TCP connection needs to have an average loss of maximally 0.03% to achieve an average bandwidth share equal to the fair one. Actually, as we did not consider the queuing delay and that the timeout value should be higher than the round trip delay, the actual loss value should be even smaller. However, the RTCP packets only include a field of 8 bits for the loss value. That is, the minimum loss value that can be reported is around 0.4%. Losses lower than this value are ignored and the LDA controlled sender would thus assume a no loss situation and further increase its transmission rate.

  Using larger fields for representing the loss values in the RTCP messages would reduce the effects of this problem. We have tested such a situation by repeating the same simulation with nine TCP and nine LDA flows competing over a link of 10 Mb/s, τ set to 1 second and the queuing delay (τ_q) set to 0.1 seconds. For the case of the limited loss fields the TCP connections only received a bandwidth share of 1.7 Mb/s of the available bandwidth. With loss fields of 32 bits instead of 8 bits the TCP traffic received a share of 2.4 Mb/s.

An alternative approach for the finer grained loss values in the RTCP messages would be to use the value of the cumulative number of lost packets (l_{cum}) included in the RTCP packets instead of the loss fraction. This value indicates the number of packets lost since the beginning of reception. While this leads to more accurate loss indications it also increases the complexity of the scheme as the sender needs to maintain more state information to actually benefit from l_{cum}.

- The second aspect that contributes to the unfair distribution is the behavior of TCP itself. Due to the large round trip delay the TCP connections can open their transmission windows to large values and thus keep more than 50 or 60 packets on the fly, i.e., packets sent but not acknowledged yet. In the case of losses several packets can get lost from the same window. In the case of Reno-TCP which was used in this study, this results in repeated rate reductions or timeouts at the TCP senders. As all the TCP connections have the same round trip delay this behavior is observed at several TCP connections at rather close time points. That is, the TCP connections get synchronized in their behavior. The synchronization and the multiple losses per round trip delay prevent the TCP connections from reaching their fair share.

  Replacing the LDA flows with TCP connections we simulated the case of 18 competing TCP connections over a link of 10 Mb/s, τ of 1 second and τ_q of 0.1 seconds. Fig. 4.19(a) shows the bandwidth share utilized by the 18 competing connections. We notice that this share oscillates and that the TCP connections can only utilize an average share of 8.4 Mb/s even though there are 10 Mb/s available. Under the same simulation parameters but replacing the TCP connections with LDA flows, 18 LDA flows manage to utilize 97% of the available bandwidth. The LDA flows suffer much less from synchronization effects than TCP due to the asynchronity of the RTCP messages that prevents the flows from reacting to the same loss simultaneously as is the case with TCP.

  Thereby, the actual share of nine TCP connections in this simulation scenario should be set to 4.2 Mb/s and not 5 Mb/s.

Taking the aspects of loss presentation, multiple losses per round trip delay and synchronization into account, we can conclude that while LDA still shows some unfairness towards TCP connections for the case of large round trip delays, this unfriendliness is, however, not as severe as the results in Tab. 4.14 to Tab. 4.17 suggest. Note additionally, that the situation described here presents an extreme case in the Internet. On the basis of data traveling at the speed of light, a round trip delay of 1 second represents a physical distance of around 300000 km which exceeds the length of almost all Internet links currently available. Such a value would, hence, usually indicate severe congestion on the

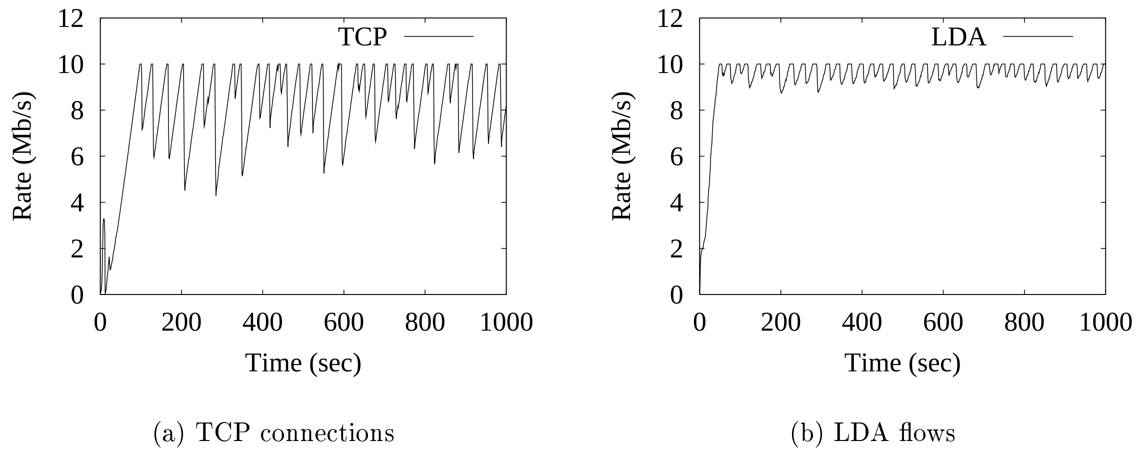


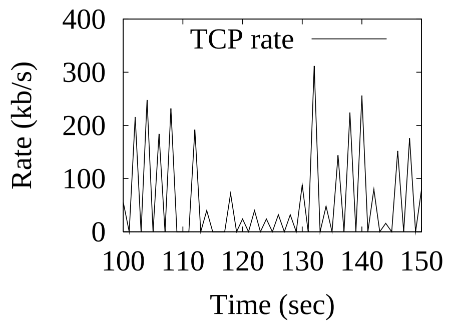
Figure 4.19: Bandwidth utilized by 18 competing flows

traversed path. Additionally, in general it is considered that delay values of more than 0.5 seconds can not be tolerated for interactive multimedia communication. Thus, the simulation parameters which resulted in this unfair behavior are rather unlikely in real networks.

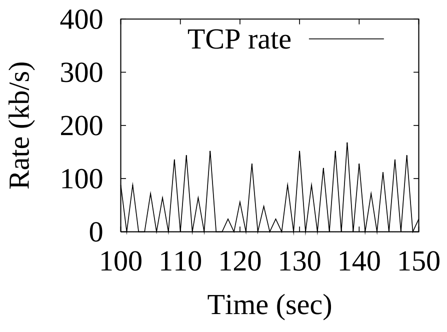
In contrast to the general tendency of increased values of F with increased round trip delay, for the case of small buffering delays of ($\tau_{\mathrm{q}} = 0.1$), large numbers of competing connections and small propagation delays some unfairness towards TCP connections can be observed as well. With 108 competing flows as in the case of Tab. 4.16, a buffer delay of 0.1 seconds translates on the average into a buffer space of one packet for each flow. The unfair bandwidth distribution in this case stems from the insufficient buffer space which causes bursty losses and the short round trip delays that cause TCP to be aggressive. This can be seen in Fig. 4.20 which shows the temporal behavior of the rate of a TCP connection for both cases of ($\tau = 0.1$ and 0.4 seconds) and ($\tau_{\mathrm{q}} = 0.1$). The rate values were measured over intervals of one seconds. We can see that for the case of ($\tau = 0.1$ seconds) the rate of the TCP connection is more often reduced to zero and hence the overall average value is lower than for the case of ($\tau = 0.4$ seconds). Note, that even though the TCP connections show a much more bursty behavior for ($\tau = 0.1$ seconds) their average bandwidth share is actually higher for the case of ($\tau = 0.4$ seconds).

4.4.2.2 Competing LDA and TCP Flows with a FIFO Router

While there is an increasing number of modern routers that support RED buffer management, most of the installed basis in the Internet supports only FIFO queuing. To test the performance of LDA in the presence of FIFO-routers, we repeated the simulations of Sec. 4.4.2.1 but with the bottleneck router using FIFO instead of RED. Tables 4.18 up to 4.21 depict the results of the simulations in terms of achieved utilization (u), friendliness (F), average standard deviation ($\overline{\sigma}$) of the average bandwidth shares of the LDA flows from the overall average rate value (r) as well as the average loss (l). Again here,



(a) $\tau = 0.1$ sec



(b) $\tau = 0.4$ sec

Figure 4.20: Effects of the round trip delay on the behavior of TCP

the presented results suggest that under varying bottleneck rates (R), queuing delays (τ_q), propagation delays (τ) and number of competing flows ($n = m = N$) a link utilization of more than 90% is achieved. The results of losses and friendliness are comparable to those reported in Sec. 4.4.2.1.

τ_q (sec)	0.10					0.50				
τ (sec)	r	$\overline{\sigma}$	F	l	u	r	$\overline{\sigma}$	F	l	u
0.1	471.2	32.4	0.7	0.006	1.0	768.7	158.7	2.1	0.004	0.99
0.2	544.2	132.2	0.95	0.005	0.98	803.25	144.5	2.5	0.005	0.99
0.4	735.09	183.1	2.04	0.004	0.97	1201.4	143.8	3.79	0.004	0.89
1.0	664.64	30.00	4.84	0.004	0.94	1278.2	47.2	5.4	0.005	0.87

Table 4.18: Achieved average rate and fairness for LDA with 9 TCP and 9 LDA competing flows and R set to 10 Mb/s for a FIFO queue

τ_q (sec)	0.10					0.50				
τ (sec)	r	$\overline{\sigma}$	F	l	u	r	$\overline{\sigma}$	F	l	u
0.1	172.97	5.43	0.83	0.030	1.00	174.45	16.08	0.87	0.012	1.00
0.2	161.48	7.01	0.74	0.021	1.00	181.82	15.12	0.94	0.011	0.99
0.4	165.60	11.35	0.80	0.014	0.99	231.09	25.68	1.35	0.011	0.94
1.0	284.83	39.32	3.43	0.006	0.97	326.84	17.88	2.92	0.015	0.90

Table 4.19: Achieved average rate and fairness for LDA with 27 TCP and 27 LDA competing flows and R set to 10 Mb/s for a FIFO queue

4.4.2.3 Performance of LDA in the Presence of WWW Traffic

The simulations in the previous sections assumed senders with an infinite amount of data for transmission. However, as described in Sec. 2.3 a considerable amount of the traffic in

τ_q (sec)	0.10					0.50				
τ (sec)	r	$\overline{\sigma}$	F	l	u	r	$\overline{\sigma}$	F	l	u
0.1	103.21	3.02	1.08	0.070	0.99	86.18	5.23	0.83	0.030	0.99
0.2	89.97	3.59	0.86	0.051	0.99	87.31	6.38	0.85	0.028	0.99
0.4	85.22	4.09	0.81	0.034	0.99	115.90	27.29	1.23	0.020	0.88
1.0	121.53	12.89	1.88	0.009	0.74	140.47	10.80	2.00	0.027	0.91

Table 4.20: Achieved average rate and fairness for LDA with 54 TCP and 54 LDA competing flows and R set to 10 Mb/s for a FIFO queue

τ_q (sec)	0.10					0.50				
τ (sec)	r	$\overline{\sigma}$	F	l	u	r	$\overline{\sigma}$	F	l	u
0.1	77.89	2.45	1.32	0.103	0.99	59.86	3.10	0.87	0.050	0.99
0.2	66.00	2.70	0.98	0.082	0.99	59.28	4.01	0.86	0.045	0.99
0.4	60.71	2.89	0.88	0.056	0.99	84.19	7.08	1.11	0.029	0.94
1.0	79.39	5.88	1.72	0.015	0.83	84.58	6.40	1.65	0.039	0.93

Table 4.21: Achieved average rate and fairness for LDA with 81 TCP and 81 LDA competing flows and R set to 10 Mb/s for a FIFO queue

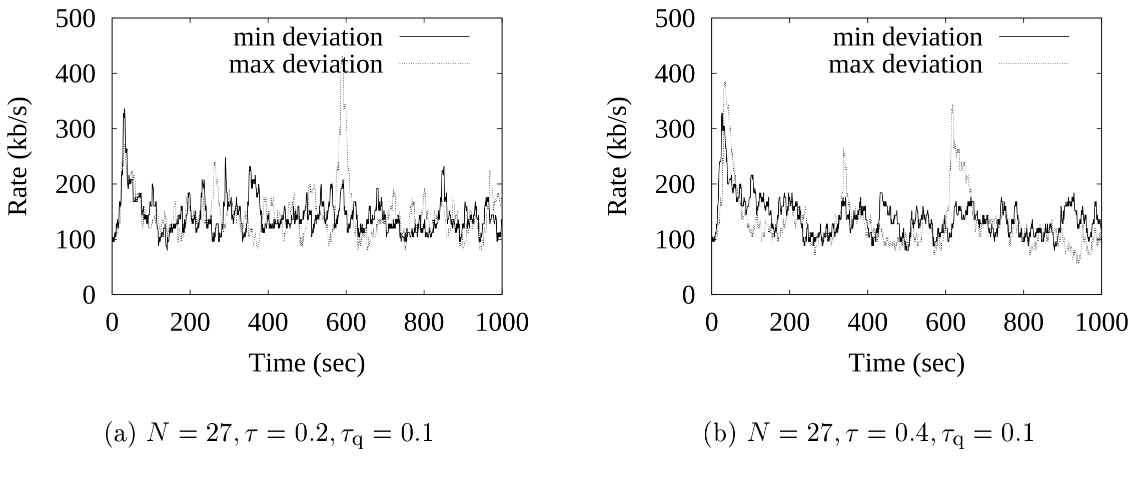
the Internet currently consists of small bursty WWW connections. To test the effects of the presence of bursty traffic on the performance of LDA we used the topology depicted in Fig. 2.5 with ($m = n = k = 27$) and the bottleneck bandwidth set to 10 Mb/s. The router used RED buffer management.

Tab. 4.22 presents the simulation results with varying round trip delays (τ) and buffering delays (τ_q). LDA achieves acceptable friendliness values except for the case of ($\tau = 1$ second) which is a result of the coarse granularity of the RTCP loss presentation. We notice, however, that the average utilization value is lower than that achieved for the case of persistent traffic only. The short bursts of the WWW traffic result in high loss peaks and thus large rate reductions on behalf of the TCP and LDA flows. This oscillatory behavior has then the effect of a reduced utilization.

τ_q (sec)	0.10					0.50				
τ (sec)	r	$\overline{\sigma}$	F	l	u	r	$\overline{\sigma}$	F	l	u
0.1	155.95	4.07	0.85	0.046	0.87	148.59	7.30	0.82	0.019	0.87
0.2	134.74	4.91	0.72	0.033	0.84	156.35	10.34	0.88	0.015	0.88
0.4	134.09	8.56	0.78	0.021	0.80	168.66	15.12	1.02	0.012	0.88
1.0	206.07	16.89	1.83	0.014	0.84	230.78	24.30	2.04	0.011	0.90

Table 4.22: Achieved average rate and fairness for LDA with 27 TCP, 27 LDA and 27 WWW servers competing flows and R set to 10 Mb/s

Fig. 4.21 describe the bandwidth share obtained by the LDA flows with the maximum and minimum standard deviation (σ). σ indicates the deviation of the temporal rate values of those flows from the average rate of the flow over the entire simulation time minus the transient time which was approximated in this study to 200 seconds. The temporal values were measured over intervals of one second. We notice that while the LDA flows show some oscillations, these oscillations are only around ±30% of the average values and occur on a slow time scale.



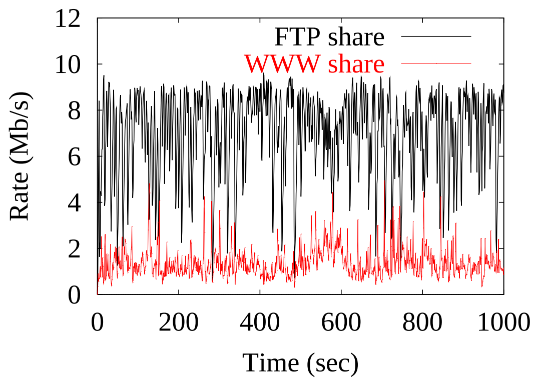
(a) $N = 27, \tau = 0.2, \tau_q = 0.1$ (b) $N = 27, \tau = 0.4, \tau_q = 0.1$

Figure 4.21: Temporal behavior of LDA

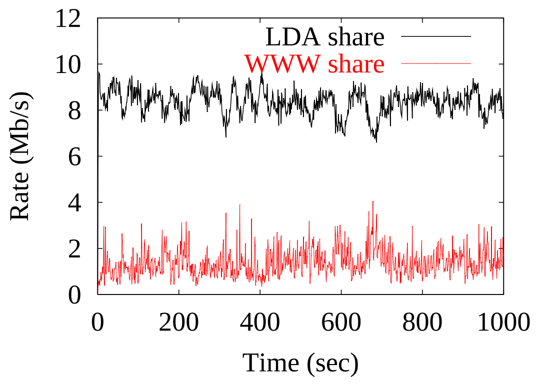
In the previous sections, see Sec. 4.4.2.2 and 4.4.2.1, we only considered the interaction between LDA flows and long lived TCP connections. That is, TCP connections running over the entire simulation time. For the case of long lived connections we aimed at achieving an equal bandwidth distribution among the LDA and TCP traffic. For the case of short lived connections, however, those performance metrics can not be used. Short lived connections do not carry enough data in order to fully utilize a bandwidth share similar to that of a long lived TCP connection. With the WWW server model we are using as described in Sec. 2.3 the average connection carries about 20 packets. Hence, those short lived TCP connections will usually send their data in a few round trip times. To test the effects of introducing LDA flows to a network inhibited by short lived TCP connections we compare the bandwidth share those short lived connections can assume when competing with the LDA flows on the one hand and when competing with long lived TCP connections on the other. In this case, we expect that the bandwidth share of the short lived TCP connections should be in both cases rather similar.

For the simulation topology depicted in Fig. 2.5 with a RED router, round trip propagation delay (τ) of 0.4 seconds, queuing delay of 0.1 seconds and a link bandwidth of 10 Mb/s we ran two simulations with 27 FTP connections and 27 WWW servers in the first case and 27 LDA flow and 27 WWW servers in the second. As described in Sec. 2.3, the WWW servers were modeled as on-off processes with the on period lasting for the time needed by a TCP sender to transmit a number of packets drawn from a Pareto distribution with the factor of 1.1 and a mean of 20 packets and the off period lasting for a time drawn from a Pareto distribution with a factor of 1.8 and a mean of 0.5 seconds.

Fig. 4.22 shows the bandwidth distribution for the case of WWW traffic competing with LDA flows on the one hand and with long lived TCP connections, i.e., FTP con-



(a) FTP and WWW



(b) LDA and WWW

Figure 4.22: Bandwidth sharing for the case of WWW traffic competing with FTP or LDA flows

nections on the other. While the LDA flows receive a much higher bandwidth share than the WWW traffic, the LDA share is similar to that of the TCP share under the same conditions. This suggests that adding LDA flows to the network does not affect the performance of the WWW traffic under the simulated situation any different than long lived TCP connections do.

4.5 Measuring the Performance of LDA over the Internet

The simulation results in Sec. 4.4.2 suggest the efficiency of LDA in reducing network congestion while maintaining a high network utilization level as well as its friendliness towards competing TCP connections over a wide range of parameters. In this part of the work, we investigate the performance of LDA when used over the wide area Internet.

For the purpose of testing LDA we conducted several measurements on different parts of the Internet. Each measurement consisted of a host sending data packets over a TCP connection to some other destination as fast as it could. Simultaneously, the host sent UDP packets to the same destination with the transmission rate determined using LDA. Each measurement was done several times over different times of the day.

The host names and domains as well as their operating systems and locations are listed in Tab. 4.23. Each measurement consisted of sending 10000 packets on both the TCP and UDP flows. The packet size was held constant to 1000 bytes which is a size often used in video conferencing applications. The initial additive increase rate ($\dot{R}_{\text{ai}}$) was set to 5 kb/s and the initial transmission rate of the UDP flows was set to 10 packets/s. For the case of transmitting data over high-capacity links with no observable losses LDA would increase the transmission rate up to the link capacity. As sending at a very high rate might lead

Host name	Domain	Operating System	Location
donald	fokus.gmd.de	SunOS 5.5	Berlin
verba	stu.neva.ru	SunOS 5.6	St. Petersburg
systems	seas.upenn.edu	SunOS 5.5	U. Pennsylvania
ale	icsi.berkeley.edu	SunOS 5.6	Berkeley, CA.

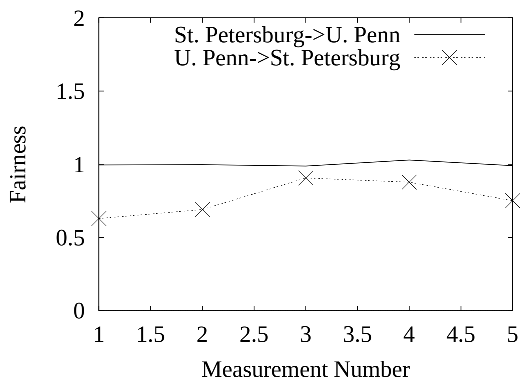
Table 4.23: Hosts used in the experimental tests

to over utilizing the processing power of the host, the maximum transmission rate of the UDP flow was limited to 800 kb/s. The friendliness factor (F) of LDA is determined here as in Eqn. 3.4. To obtain a detailed view of the performance of LDA we collected information about the sent and received RTP and TCP data in intervals of one second. Additionally, we collected the loss, round trip delay and bottleneck bandwidth information carried by the RTCP protocol. To estimate the average bottleneck bandwidth we relied on an approach similar to that used in the BPROBE tool [29] for filtering out incorrect estimates. That is, similar bottleneck bandwidth values estimated by the receiver using Eqn. 2.10 were clustered into intervals and the average of the interval with the highest number of estimates was chosen. The estimated value was then sent back to the sender with the receiver reports.

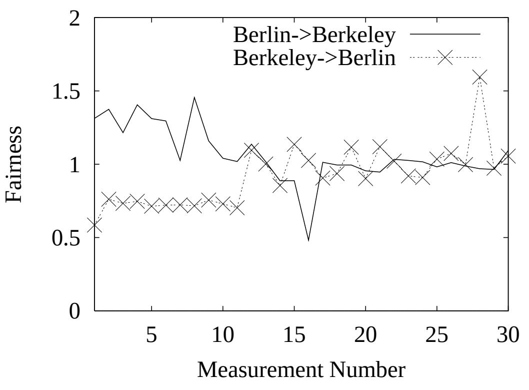
Obviously, this approach has lots of drawbacks. We did not include the time between the transmission of two probe packets. But, as we sent the probe packets at the sender's network access speed which in our case was 10 Mb/s we can usually ignore this time. Also, we did not consider packet drops or competing traffic. However, testing this approach on different Internet connections we achieved results comparable to those estimated by a more complicated tool such as the PATHCHAR tool [43].

Fig. 4.23 show the friendliness results as measured over different links in Europe and the States. Fig. 4.24 depict the average loss values observed during these measurements.

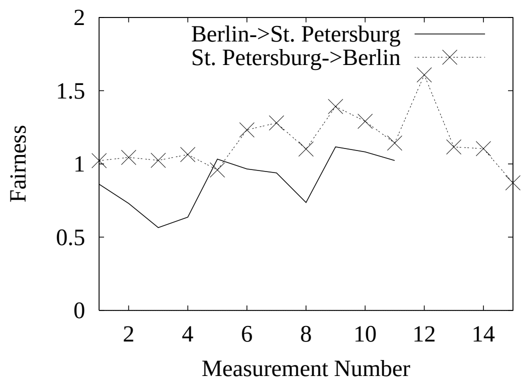
The fairness results depicted in Fig. 4.23 show great variations among the different flows and even on the different directions of the same flow. The links between St. Petersburg and Berlin as well as St. Petersburg and U. Penn. are rather lossy in both directions. Under these conditions the measured friendliness index (F) varies between 0.6 and 1.4 with most of the measured values in the range of 0.8 and 1.2. These results are rather similar to the results achieved using the simulation model in Sec. 4.4.2 and indicate the TCP-friendliness of LDA. The results depicted in Fig. 4.23(d) are, however, contradictory. In the direction form Berlin to U. Penn. we have a friendliness factor of around 0.8 on the average which means that the LDA controlled flow actually receives a smaller share of the bandwidth than the competing TCP connection. The measurements on the opposite direction indicate, however, that the LDA controlled flow receives four times as much bandwidth as the competing TCP connection. Actually, the LDA controlled flow manages to achieve the maximum transmission rate and to stay at this level. While these results sound contradictory on the first sight, they actually resulted from the asymmetry of the Internet link between Europe and the States. Looking at the loss results depicted in Fig. 4.24(d) reveals that while the traffic from Berlin to U. Penn. suffered on the average from losses below 1%, the traffic sent in the other direction had an



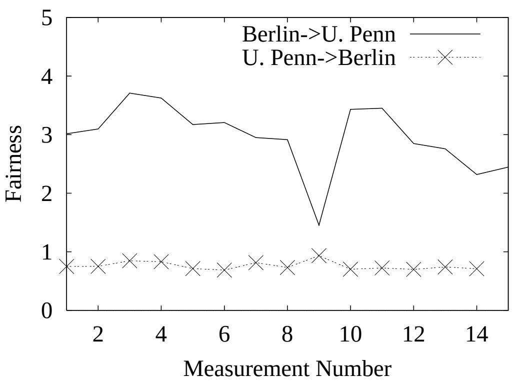
(a) U. Penn.$\Longleftrightarrow$ St. Petersburg



(b) Berlin$\Longleftrightarrow$ Berkeley



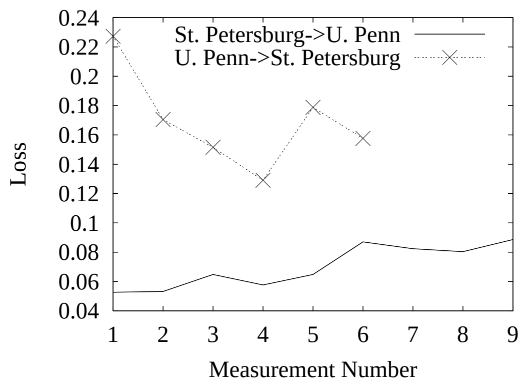
(c) Berlin$\Longleftrightarrow$ St. Petersburg



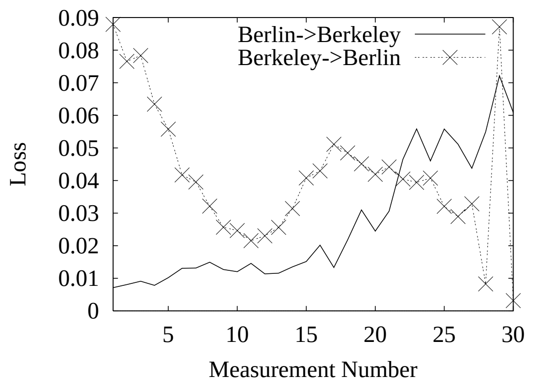
(d) Berlin$\Longleftrightarrow$ U. Penn.

Figure 4.23: TCP friendliness measured over different Internet paths

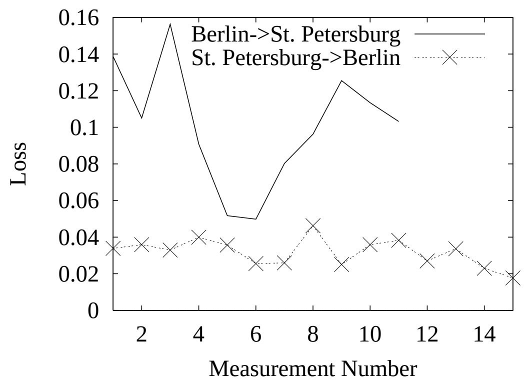
average loss of more than 10%. Fig. 4.25 shows a detailed snapshot of the measurement labeled 2 in Fig. 4.23(d) and displays the transmission rate of both the TCP connection and the LDA controlled flow and the losses measured over intervals of one second in both directions on the link connecting U. Penn. and Berlin. In Fig. 4.25(b) we notice that while in the direction from Berlin to U. Penn. no losses were observed during the entire measurement period the traffic from U. Penn. to Berlin faced losses ranging from 5% up to 25% during the same period. This asymmetry leads to the well known *ack compression* problem [203, 88]. On symmetric links TCP acknowledgment packets arrive at the sender with a similar rate to the arrival rate of the data packets at the receiver. During the congestion avoidance phase, each arriving acknowledgment clocks one or two new packets out at the sender and leads to an increase in the congestion window (CWND) by $\frac{1}{\mathrm{CWND}}$ each round trip delay. For the case of a slower or lossy back link, acknowledgments might arrive in clusters or might get lost. In the worst case, loosing several acknowledgment packets in a row might cause a timeout at the sender which leads to a severe reduction of the congestion window and the sender retransmitting packets that have already reached



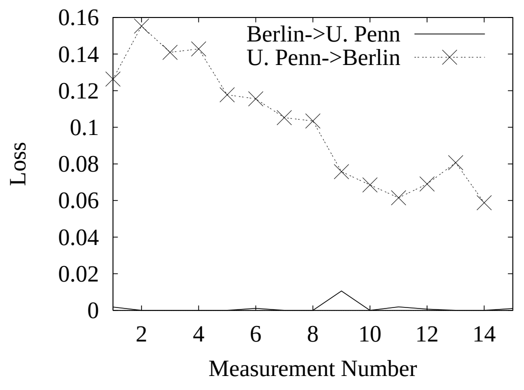
(a) U. Penn.$\Longleftrightarrow$ St. Petersburg



(b) Berlin$\Longleftrightarrow$ Berkeley



(c) Berlin$\Longleftrightarrow$ St. Petersburg

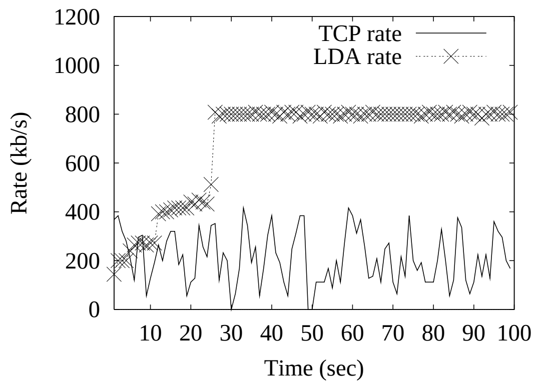


(d) Berlin$\Longleftrightarrow$ U. Penn.

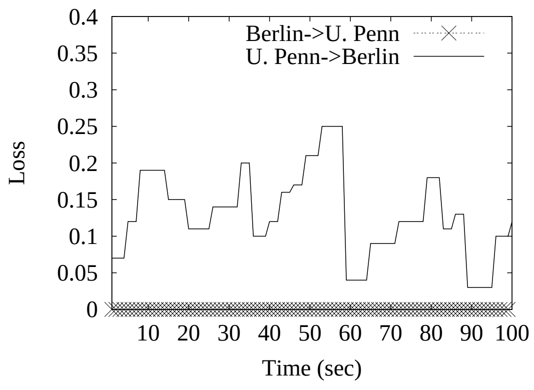
Figure 4.24: Losses measured over different Internet paths

the receiver. In any case, the clustering or loss of acknowledgments can result in idle states at the sender. That is, after sending a complete window worth of packets, for example from packet (P_1) up to packet (P_y), the sender needs to wait for an acknowledgment for packet (P_1). If the acknowledgment for this packet was lost the sender will only be able to start sending data after receiving the acknowledgment for packet ($P_x : y \geq x > 1$). Further, after receiving the acknowledgment for (P_x) the sender can send up to x packets instead of one or two at once at the access speed of the sending station. Sending a large burst of back to back data packets increases the probability of one of those packets being dropped as the speed of the transmission and the number of packets might exceed the capacities of some router on the path to the receiver.

So while the transmission rate of UDP flows is adapted only to the losses on the way from the sender to the receiver, the TCP connections' transmission behavior is affected by the loss conditions on the direction from the receiver to the sender as well. Thus, in this situation setting the transmission rate of the UDP senders exactly to the equivalent TCP rate would be ineffective. The transmission rate of the UDP sender would be artificially restricted to a level that actually leads to underutilization of the network and might not fulfill the needs of the user.



(a) Bandwidth distribution



(b) Loss

Figure 4.25: Bandwidth distribution and losses measured on both directions from Berlin to U. Penn.

Adaptation schemes that adjust the transmission rate based on acknowledgment packets from the receivers such as [130, 146, 74] have the same problem as TCP and can not benefit from the available network resources appropriately. Additionally, note that during the entire observation period in Fig. 4.24 no losses were measured on the link from the sender to the receiver. Thus, in such a situation relying solely on the TCP model of Eqn. 2.6 is not appropriate for determining the transmission rate and the approach discussed in [61] needs, hence, to be extended to handle this situation.

Fig. 4.26 shows the goodput, measured in intervals of 2 seconds, of the competing TCP and LDA streams during a period of 200 seconds of the measurement shown as point 18 in Fig. 4.23(b). LDA shows here a less oscillatory behavior than TCP and has in general a comparable rate to that of TCP.

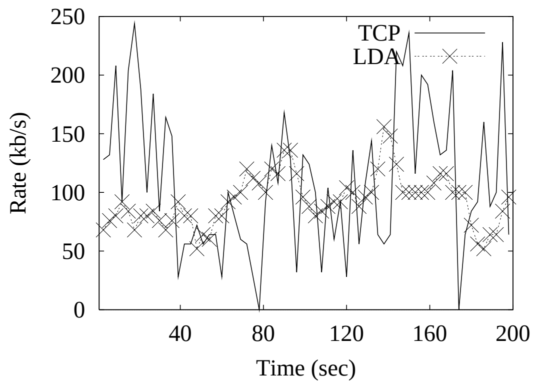


Figure 4.26: Temporal behavior of competing TCP and LDA streams

4.6 Comparison of the Performance of LDA to Other Congestion control Schemes

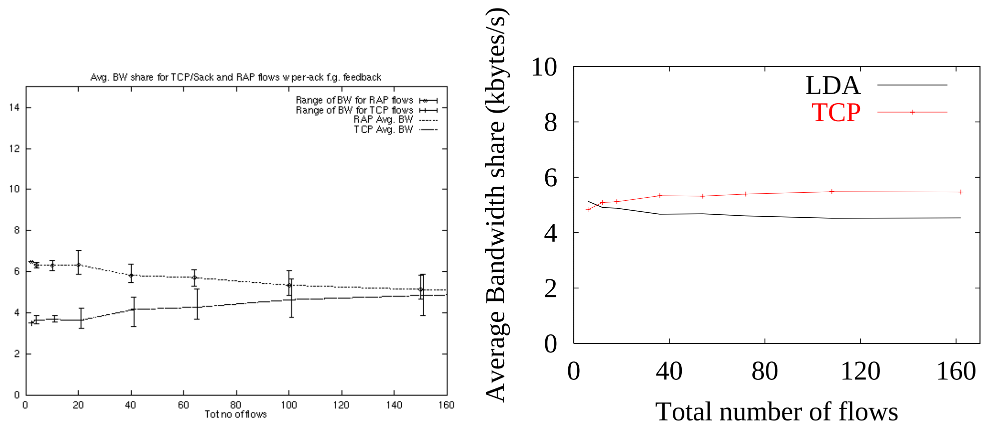
In this part of the work, we compare between the performance of LDA and a row of recent proposals for TCP-friendly congestion control. In order to cover a wide range of possible approaches for TCP-friendly adaptation we compare LDA to a rate-based scheme and a scheme based on an analytical model of TCP. For comparing the schemes, we picked out some representative test cases as were described in the papers presenting those algorithms. We re-simulated those cases in our simulation environment and compared the achieved results using LDA with the results achieved by the other schemes as were reported by their authors. This approach reduces possible errors in the comparisons due to misinterpretations or wrong implementation of the algorithms.

4.6.1 Comparison of LDA with a Rate-Based TCP-Friendly Adaptation Scheme

With the rate adaptation protocol (RAP) [146], already briefly described in Sec. 3.1.2.2, the receiver acknowledges the reception of each incoming packet. Based on those acknowledgments the sender can estimate the round trip delay and losses. In the absence of packet losses the sender increases its transmission rate by an additive amount determined using the estimated round trip delay. Additionally, the transmission rate is subject to fine grain adaptation related to variations of the short-term average round trip delay compared to the long-term average round trip delay. After detecting a packet loss, the rate is reduced by half in a similar manner to TCP.

For testing the performance of RAP Rejai et al. used in [146] a simulation topology similar to that depicted in Fig. 2.5 with N TCP connections sharing a single bottleneck with N UDP flows. The round trip delay (τ) was set to 0.04 seconds and the bottleneck bandwidth (R) was set to ($2 \times N \times 40$ kb/s). The router used FIFO buffer management with the buffer length set to ($R \times \tau \times 4$). The packet size was set to 100 bytes. To avoid synchronization among the flows, each transmitted packet was delayed randomly by an amount varying between 0 and the bottleneck service time. In [146] SACK-TCP connections were used. As our simulation environment does not support SACK TCP we referred to using Reno-TCP which delivers similar results in general but might show a lower performance for the case of bursty losses.

Fig. 4.27 depicts the results of the average bandwidth share for the adaptive flows and the TCP connections. We can observe that even though RAP requires a higher control overhead due to the frequent acknowledgments, equal bandwidth shares are only reached for higher numbers of competing flows. For a low number of competing flows the RAP flows actually receive a 50% higher bandwidth share than the TCP connections. Additionally, note that Fig. 4.27(a) describes the results reached with SACK-TCP which is more robust to multiple losses than Reno-TCP which we have used. Actually, the authors of [146] report that with Reno-TCP the fairness results of RAP are slightly worse. With LDA, the TCP connections receive on the average a bandwidth share of around 5.4 kbytes/s compared to 4.5 kbytes/s for the LDA flows. Hence, LDA is under



(a) RAP (b) LDA

Figure 4.27: Bandwidth distribution with RAP and LDA

this configuration more conservative than TCP.

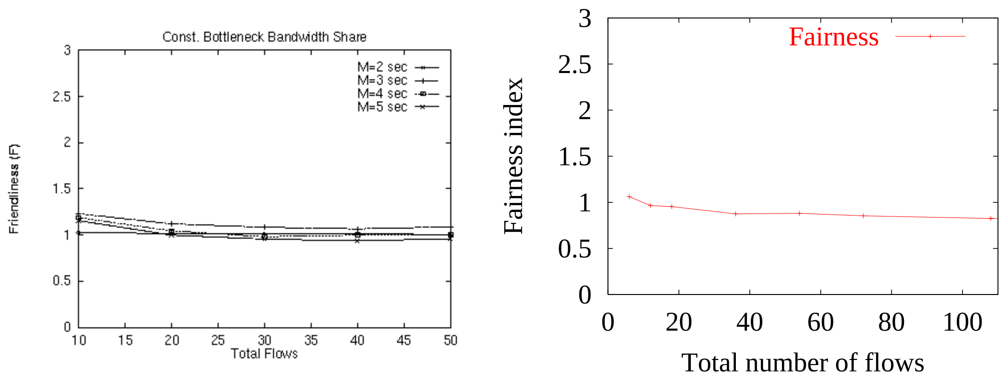
4.6.2 Comparison of LDA with an Equation-Based TCP-Friendly Adaptation Scheme

Using the analytical model of TCP in Eqn. 2.6 Padhye et al. present in [130] a scheme called TCP-friendly rate control protocol (TFRCP). With TFRCP the sender estimates the round trip delay and losses based on the receiver's acknowledgments. In the case of losses, the sender restricts its transmission rate to the equivalent TCP rate calculated using Eqn. 2.6 otherwise the transmission rate is doubled. The sender updates its rate periodically in intervals of M. In [130] different results were reported for adaptation intervals between two and five seconds.

Similar to our previous investigations a simple test topology was used in [130] with N TCP connections sharing a single bottleneck with N UDP flows. The round trip delay (τ) was set to 0.04 seconds and the packet size was set to 100 bytes. The router used FIFO buffer management with the buffer length set to $(4 \times R \times \tau)$.

Fig. 4.28 depicts the fairness results of a test setting with the bottleneck bandwidth set to $(2 \times N \times 40$ kb/s) for a varying number of flows. The fairness index is determined as the average goodput of the adaptive flows to the average goodput of the TCP connections. Both LDA and TFRCP achieve in this case a fairness index of around one indicating a high degree of TCP-friendliness. With TFRCP this is especially evident when an adaptation interval (M) of two seconds is used. For higher values of M TFRCP is especially for the case of fewer competing flows more aggressive than TCP.

Fig. 4.29 depicts the fairness results for the case of a constant bottleneck bandwidth of 1.5 Mb/s. Depending on the chosen adaptation interval, TFRCP achieves different fairness values ranging from 0.7 to around one. For higher numbers of competing flows, LDA achieves a fairness index of around one as well indicating its TCP-friendliness in



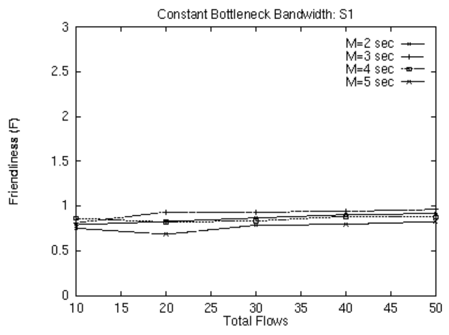
(a) TFRCP (b) LDA

Figure 4.28: Fairness index with TFRCP and LDA for a constant bandwidth share per flow

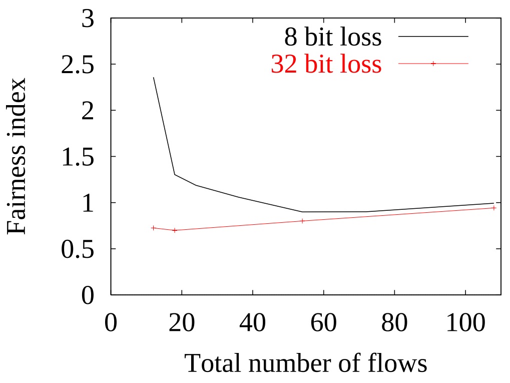
cases of higher network loads. However, for the case of a lower number of competing flows a fairness factor higher than one is achieved. This is especially evident for the case of six LDA flows competing with six TCP connections. In this case, a fairness index of 2.3 is achieved indicating that the LDA flows receive on the average twice as much bandwidth as the TCP connections. However, this is not caused by the adaptation algorithm itself but by the way the loss information are transported in RTP. The receiver reports use an eight bit field for the loss information and hence the smallest loss information that can be indicated is approximately 0.004. However, under the used topology with 0.04 seconds round trip delay and a bottleneck of 1.5 Mb/s to be divided among 12 competing flows, using Eqn. 2.6 indicates that the average loss rate each flow needs to suffer in order for it to get a bandwidth share of ($\frac{1.5}{12} = 0.125$ Mb/s) is around 0.002. In this case, the RTP receivers approximate the loss value down to 0 and the RTP flows can increase their transmission rate in situations where they are supposed to reduce it. Fig. 4.29(b) depicts the fairness results achieved when using a 32 bit field in the receiver reports for loss indication instead of eight bits. In this case, we can observe that LDA is rather conservative and achieves for the case of smaller numbers of competing flows a fairness index of less than one. For the case of higher numbers of competing flows the behavior of LDA is identical for both loss representations.

4.7 Summary

In this chapter, we first investigated different possibilities for realizing adaptive TCP-friendly congestion control and presented an algorithm called the loss-delay based adaptation algorithm (LDA) for estimating the TCP-friendly bandwidth share a flow can consume. The estimation is based on information collected by the receiver about the loss, delay and bottleneck bandwidth of the path traversed by the data flow. The collected



(a) TFRCP



(b) LDA

Figure 4.29: Fairness index with TFRCP and LDA for a constant bottleneck bandwidth

information are then sent back to the sender using the control part of RTP. Performance results obtained by conducting simulations and measurements over different topologies with varying parameters suggest the TCP-friendliness of LDA over a wide range parameters. Additionally, comparing LDA to other currently proposed TCP-friendly schemes reveals that even though LDA uses a low control overhead of one control message every a few seconds it still displays comparable if not superior behavior to those schemes which exchange control information in short periods of around one round trip delay. Actually, this infrequency of the control messages, allows LDA to display a more robust behavior for the case of asynchronous links where lossy links might lead to high delays and losses of feedback information. While the infrequency of the RTP control information does not impose significant constraints on the behavior of LDA in terms of TCP-friendliness the simulation results presented in Sec. 4.4 suggest that the granularity of the loss information carried with RTP might lead to unfair bandwidth distribution between LDA and TCP flows. This is especially evident for the case of large round trip delays and low losses. In this case, losses are often approximated down to zero allowing an LDA sender to increase its transmission rate at a time when a competing TCP connection actually reduces its. Hence, to further improve the behavior of LDA, RTP needs to be enhanced to allow for a more granular loss representation.

Chapter 5

Constrained Congestion Control for Multimedia Communication

Up till now we assumed that the sender can adjust its transmission rate in accordance with the values determined by the congestion control scheme with arbitrary granularity and with no boundaries as to the maximum or minimum bandwidth values. However, in reality multimedia encoders often impose strict limitations as to the possible adaptation steps or the minimum bandwidth requirements below which the sent data is meaningless to the receiver. Additionally, there are some restrictions arising from the user himself with regard to the acceptable variations in the perceived QoS over a time interval.

To investigate the effects of such restrictions on the adaptation process we present in Sec. 5.1 a model of a generic multimedia source and describe some constraints such a model might impose on the adaptation process. While this model does not represent a specific audio or video compression style, a communication scenario or particular user preferences, the model can be tuned through different parameters to describe a wide range of possible characteristics of multimedia sources and user preferences. Based on this model, we describe in Sec. 5.2 a general framework called constrained TCP-friendly adaptation framework (CTFAF) for assisting rate adaptation algorithms in incorporating various constraints in the adaptation process on the one hand and retaining fair bandwidth distribution characteristics required from the algorithms on the other. The effects of enhancing an adaptation algorithm such as LDA to accommodate user and media constraints is then investigated in terms of bandwidth utilization, stability and TCP-friendliness in Sec. 5.3.

5.1 A Generic Model for Multimedia Communication

In designing an adaptation algorithm, we need to take the constraints imposed by specific characteristics of the user and transmitted multimedia content into account. In this section, we present a generic model that describes possible constraints imposed by the user, compression algorithm and data content on the adaptation process. This model considers the following constraints:

Fixed limits: The quality of a multimedia stream can usually be improved by increasing the bandwidth share of the stream. However, above a certain limit ($r_{\max}$) no noticeable improvements in the QoS will be observed anymore. Additionally, reducing the bandwidth share of the stream below some minimal rate ($r_{\min}$) would redeem the transported data useless by the receiver. The values of those maximum and minimum values depend highly on the used compression styles, the users and the data content. While transmitting a talking heads kind of video conference at a frame rate of only a few frames per second might be acceptable, transmitting a live sports event requires obviously a high frame rate in order to be able to keep track of all instantaneous and rapid changes in the play field.

Granular adaptation: Depending on the used compression scheme and data content, a data source might only be able to change its transmission rate in steps of $S_{\text{adaptation}}$. Taking the example of an MPEG video stream, changing the transmission rate can be achieved by either changing the transmission frame rate or the quantization level. In this case, the adaptation can only be realized in steps of ($S_{\text{adaptation}}$) at the granularity of the discrete quantization steps or the size of a video frame. For the case of audio communication, adaptation can often only be realized by changing the used compression style which yields fixed adaptation steps as well.

Stable presentation: With TCP the transmission rate is changed in the order of once every round trip time in possibly large steps. Applying such an adaptation behavior to multimedia flows would result in a highly fluctuating perceived quality that can be annoying to the user. This can be observed for example when rapidly changing the frame rate for the case of video communication or continuously changing the used audio compression scheme.

To provide the user with a stable perceived quality, the adaptation scheme needs to limit the maximum changes ($\delta_{\text{adaptation}}$) as well the rate of changes of the transmission rate of a multimedia stream. This allows for smooth changes in the transmission rate and a stable perceived QoS.

Loss tolerance: Depending on the transferred content, the user and the used compression schemes, some data losses might be tolerated during a multimedia communication session. For example some compressions schemes already include loss recovery mechanisms for hiding data losses form the user. As another example, while users listening to an audio stream carrying music would require a very low loss rate, users involved in an interactive audio conference would be more tolerant to losses. Again here, the tolerated loss depends not only on the loss rate and distribution pattern but also on the fluency of the users in the used communication language for example.

Note, that this model only presents a subset of all the possible constraints that need to be considered when designing an adaptation scheme. Another example for such a constraint results from the case that changing some compression parameters must be followed by a recovery period to allow the senders and receivers to synchronize the encoding and decoding process in accordance with the new parameters.

5.2 A General Framework for Rate Adaptation for Constrained Multimedia Flows

When adapting the rate of a multimedia source in accordance with the network congestion state, one has to take the constraints imposed by this source as described by the generic model of Sec. 5.1 into account and at the same time achieve TCP-friendly congestion control.

In this section, we present a general framework for adapting the transmission rate of constrained multimedia sources. With this framework, called the constrained TCP-friendly adaptation framework (CTFAF), the source uses a TCP-friendly adaptation algorithm for determining the bandwidth share of the multimedia flow. However, instead of adjusting the transmission rate of the source solely based on the rate determined by the control algorithm, the constraints imposed by the specific characteristics of the multimedia source are taken into account as well. This means that at some time points, the sender has to reduce its transmission rate by an amount less than determined by the adaptation scheme. To take such divergence from the TCP-friendly behavior into account the sender sums the difference between the change in bandwidth values as determined by the adaptation scheme and the changes allowed by the multimedia source into a variable called virtual bandwidth (B_{virtual}). A negative value of B_{virtual} indicates that the source was too aggressive in the past. Therefore, to compensate this aggressive behavior the sender should reduce its rate by the maximum allowed value by the multimedia model every time a rate reduction is requested by the adaptation scheme. Finally, to avoid the case where an extreme value of B_{virtual} affects the adaptation behavior of the multimedia source over long time intervals, B_{virtual} is periodically reset to zero and the transmission rate is adjusted by an averaged value of B_{virtual}.

Considering all the possible constraints imposed by multimedia compression styles and contents as well as the users in the adaptation process requires deep knowledge of the compression style as well as the content to be transmitted. Additionally, the minimum and maximum acceptable transmission rates result from the content and the communication scenario. Finally, the acceptable rate of changing the QoS depends also on the content as well as the user himself.

As a simplification of this problem we consider in this part of the work a generic encoder with the following characteristics:

- A minimum bandwidth requirement of r_{min} and a maximum one of r_{max}.

- The adaptation itself can only be realized in steps of $(m \times S_{\text{adaptation}})$.

- To accommodate the requirement of a stable perceived QoS the changing rate $(\delta_{\text{adaptation}} = \frac{\Delta r}{\Delta t})$ of the transmission rate of a sender should not be increased or decreased by more than $(n \times S_{\text{adaptation}} \leq \delta_{\text{adaptation}})$ during a time interval $(T_{\text{adaptation}})$, whereas $T_{\text{adaptation}}$ can be considered as the time between two adaptation points, i.e., the time points at which the sender executes the adaptation algorithm and adapts its transmission rate.

- A sender does not need to adjust its transmission rate if losses were below l_{allowed} or its transmission rate would go beneath r_{min} or above r_{max}.

As a generic adaptation algorithm we consider here a scheme that reduces the transmission rate of the sender by an amount X_{r} during congestion periods and increases it by an amount of X_{i} during network underload situations. The adaptation actions are triggered every $T_{\text{adaptation}}$.

Now imposing constraints on the adaptation process means that during some time intervals the transmission rate of some flow can not be reduced even though losses are measured or the rate can not be increased even though the network is underloaded. To be able to integrate these constraints with the adaptation algorithm and still allow for a fair bandwidth distribution we incorporate the concept of virtual bandwidth share (B_{virtual}) with the adaptation process. A positive B_{virtual} represents the amount of bandwidth the flow should have acquired based on the calculations of some adaptation scheme but did not use due to the constraints of the multimedia source. As the unused resources were consumed by competing flows an average equal bandwidth distribution can then be achieved if, in the future, the flow consumed a bandwidth share larger than the rate calculated by the adaptation algorithm ($r_{\text{calculated}}$). A negative value indicates the amount of bandwidth used in excess of the amount calculated by the adaptation scheme and hence that the flow was too aggressive in the past and should use a transmission rate lower than $r_{\text{calculated}}$ when possible. While this approach does not guarantee fair bandwidth distribution at all times it allows for a fair distribution over long time periods.

With such an enhancement to an adaptation algorithm, the increase and decrease behavior of the control algorithm is influenced by the value of B_{virtual} in addition to the limitations imposed by the user or the data source as well as the network congestion state.

B_{virtual} is initialized to 0 at the beginning of the transmission process and then adjusted as follows:

- If the sender is allowed to increase its transmission rate by an amount $\{X_{\text{i}} : X_{\text{i}} > \delta_{\text{adaptation}}\}$ then the sender increases its transmission rate by $\delta_{\text{adaptation}}$ and B_{virtual} is set to

$$B_{\text{virtual}} = B_{\text{virtual}} + X_{\text{i}} - \delta_{\text{adaptation}} \tag{5.1}$$

- If the sender needs to reduce its transmission rate by $\{X_{\text{r}} : X_{\text{r}} > \delta_{\text{adaptation}}\}$ then he only reduces its transmission rate by $\delta_{\text{adaptation}}$ and B_{virtual} is set to

$$B_{\text{virtual}} = B_{\text{virtual}} - X_{\text{r}} + \delta_{\text{adaptation}} \tag{5.2}$$

- If the sender needs to reduce its transmission rate by an amount X_{r} due to a loss (l) lower than the allowed loss value the flow can ignore or repair by himself $\{l : l < l_{\text{allowed}}\}$ or the reduction would lead to a transmission rate $\{r : r < r_{\text{min}}\}$ then he does not reduce its transmission rate and B_{virtual} is set to

$$B_{\text{virtual}} = B_{\text{virtual}} - X_{\text{r}} \tag{5.3}$$

An optimal adaptation behavior is realized with $B_{\text{virtual}} \approx 0$. To achieve such a behavior, in CTFAF, we additionally adjust the decrease and increase behavior of the adaptation algorithm to consider B_{virtual} and hence reduce the effects of temporarily unfair distribution as fast as possible.

Overload situation: For the case of network congestion the sender should reduce its current transmission rate by X_{r}.

Depending on the value of B_{virtual} the sender behavior is adjusted as follows:

$B_{\text{virtual}} > 0$**:** A positive B_{virtual} indicates that this sender was too conservative in the past. As a compensation the sender can ignore the loss situation and maintain its current transmission rate. B_{virtual} is then reduced by the amount the rate should have been reduced by:

$$B_{\text{virtual}} = B_{\text{virtual}} - X_{\text{r}} \tag{5.4}$$

$B_{\text{virtual}} < 0$**:** A negative virtual bandwidth indicates on the contrary that the sender has been too aggressive in the past. Therefore, for the case of $X_{\text{r}} < \delta_{\text{adaptation}}$ the rate is reduced by $\delta_{\text{adaptation}}$ and B_{virtual} can be increased as follows:

$$B_{\text{virtual}} = B_{\text{virtual}} + \delta_{\text{adaptation}} - X_{\text{r}} \tag{5.5}$$

Underload situation: For the case of network underload the sender would be allowed to increase its transmission rate by X_{i}. To compensate a negative virtual bandwidth the sender maintains in this case its current transmission rate. Thereby B_{virtual} is changed as follows:

$$B_{\text{virtual}} = B_{\text{virtual}} - X_{\text{i}} \tag{5.6}$$

Recent values of B_{virtual} bear more significance in terms of characterizing the friendliness or aggressiveness of a flow towards competing traffic. Hence, to avoid having some value of B_{virtual} propagating over long time periods and influencing the behavior of the adaptation scheme even though the network situation that has resulted in this value has changed considerably, we refer to resetting B_{virtual} in periods of T_{reset}. T_{reset} is several times as large as the adaptation interval ($T_{\text{adaptation}}$), i.e., the time between two adaptation actions. At the end of each resetting period (T_{reset}) the sender changes its transmission rate by an averaged B_{virtual} ($\overline{B_{\text{virtual}}}$) and resets B_{virtual} down to zero. The averaged B_{virtual} is determined as

$$\overline{B_{\text{virtual}}} = B_{\text{virtual}} \times \frac{T_{\text{adaptation}}}{T_{\text{reset}}} \tag{5.7}$$

As B_{virtual} is changed at each adaptation point, $\overline{B_{\text{virtual}}}$ represents an averaged value of all additions and subtractions done during T_{reset}.

Using a ($\overline{B_{\text{virtual}}} : \overline{B_{\text{virtual}}} > \delta_{\text{adaptation}}$) contradicts the requirement of limiting the maximum changes in the transmission rate to $\delta_{\text{adaptation}}$. However, we found this approach to be necessary in order to adjust the performance of a flow in accordance with the available resources. Without this resetting behavior, a sender transmitting data at a rate higher than the capacity of some link on the path towards the receiver and specifying a small $\delta_{\text{adaptation}}$ compared to the needed change in the transmission rate would cause high and long lasting congestion. For the case that the flow is using a bandwidth share close to the average share calculated with the adaptation scheme, $\overline{B_{\text{virtual}}}$ would have a small value and would not have considerable effects on the adaptation behavior.

Without prior knowledge of the available resources a sender might start transmitting data at a rate much higher than its fair share. If the sender additionally specified a slow maximum changing rate ($\delta_{\text{adaptation}}$) the sent flow would lead to an overload situation lasting at least till the end of the first resetting period (T_{reset}). As T_{reset} might be in the range of one or more minutes, relying solely on the resetting actions would take too long. To overcome this situation we specify that for a transient phase lasting for a maximum period of T_{init} the sender applies the adaptation scheme restricted only by the encoder's limitations and not the user specifications. That is, for the first T_{init} seconds the sender can adapt its transmission rate by any number of adaptation steps ($S_{\text{adaptation}}$) as dictated by the adaptation scheme without regard to $\delta_{\text{adaptation}}$.

Note that the proposed guidelines here are only one option for accommodating user and media constraints into the adaptation process. For example, resetting B_{virtual} periodically is only one possible approach for preventing B_{virtual} accumulated during past periods to have long lasting effects on the adaptation process. Other options might be to use weighted moving averages or requiring that flows having a negative B_{virtual} value higher than some maximum value must reduce their rate more often and by larger amounts than indicated by the adaptation algorithm and user preferences. The exact strategies to use as well as their parameters will depend on the communication scenario.

5.3 Performance Investigation of CTFAF

In this part of the work, we investigated the effects of imposing user- or encoder-based restrictions on the adaptation process in terms of buffer occupancy and bandwidth utilization. For this purpose we compared the performance of LDA without any restrictions, i.e., ($\delta_{\text{adaptation}} = \infty$) and its performance when the rate adjustments were controlled not only by network conditions but by the CTFAF guidelines described in Sec. 5.2 as well.

For investigating the performance of CTFAF we used the simulation topology depicted in Fig. 2.5 with FTP, WWW and multimedia flows competing for a bottleneck router of 10 Mb/s and ($m = n = k = 27$).

The routers used random early packet discard (RED) [55] buffer management with the minimum drop threshold set to 0.3 of the router buffer and the maximum to 0.8. The maximum queuing delay (τ_q) was set to 0.1 seconds.

Unless otherwise stated, the round trip propagation delay was set to 0.4 seconds. The initialization period (T_{init}) was set to 60 seconds. The performance of CTFAF was tested in terms of achieved average rate (r), fairness factor (F), average standard deviation (σ) and average utilization value (u). The average rate (r) was determined over the entire simulation time minus a transient time which was approximated in this case to the first 200 seconds. The fairness index (F) was determined as the ratio of the bandwidth share of the CTFAF flows to the bandwidth share consumed by the TCP connections. σ indicates the average of the deviation value of the temporal rate values of each CTFAF flow from the average rate of that flow and is used here as an indication of the stability of the estimated transmission rate. The temporal values were measured over intervals of one second.

5.3.1 Effects of the Length of the Adaptation Steps ($S_{\text{adaptation}}$) on the Performance of CTFAF

As an example for a multimedia flow we simulated the case of video flows with constant quantization steps of 2 kbits, packet sizes of 10 kbits and frame sizes of 20 kbits. We tested the case of coarse and fine granularity adaptation. Using ($S_{\text{adaptation}} = 20$ kb/s) resembles the case of changing the transmission rate by varying the number of sent video frames when using JPEG compression for example. We set $\delta_{\text{adaptation}}$ in this case to 1 and 2 frames/s, i.e., 20 and 40 kb/s. With ($S_{\text{adaptation}} = 2$ kb/s) the adaptation is more granular and simulates the case of varying the quantization or quality factor. $\delta_{\text{adaptation}}$ is set in this case to 4 kb/s. As a minimum transmission rate we used a value of 40 kb/s. The parameter settings for the video coder were chosen based on observations made during a talking heads conference with JPEG as the compression style. For another communication scenario or a different compression style we would most likely have to use a different set of parameters. Refining the video model and choosing the appropriate parameters requires detailed knowledge of the used compression style and communication scenario. Additionally, to account for the user preferences extensive subjective testing is needed. The resetting period (T_{reset}) was set to 60 seconds. From Tab. 5.1 we can observe

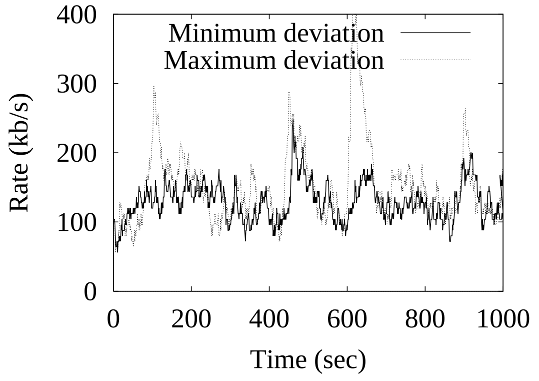
$\delta_{\text{adaptation}}$ (kb/s)	r (kb/s)	σ (kb/s)	F	u
∞	134	44	0.8	0.81
40	137	45	0.84	0.80
20	132	37	0.80	0.79
4	140	29	0.85	0.81

Table 5.1: Achieved average rate and fairness of CTFAF with different values of $\delta_{\text{adaptation}}$

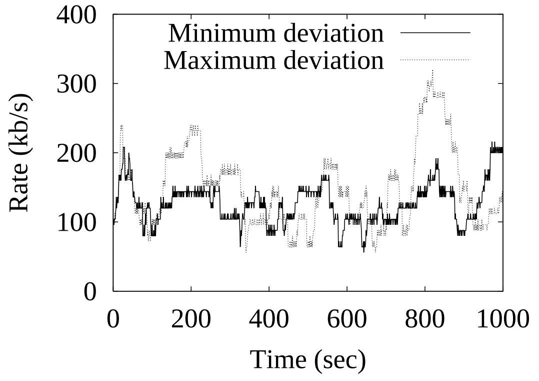
that while the standard deviation which indicates the level of oscillations of the flows is reduced for finer settings of $\delta_{\text{adaptation}}$ the utilization and fairness levels are hardly changed compared to the case of no restrictions on the adaptation ($\delta_{\text{adaptation}} = \infty$).

The reduction of oscillations is particularly evident in Fig. 5.1 which shows the temporal behavior of the flows with the largest and smallest standard deviation (σ) values. For the case of ($\delta_{\text{adaptation}} = 4$ kb/s) the flows have a much smoother shape compared to the other cases.

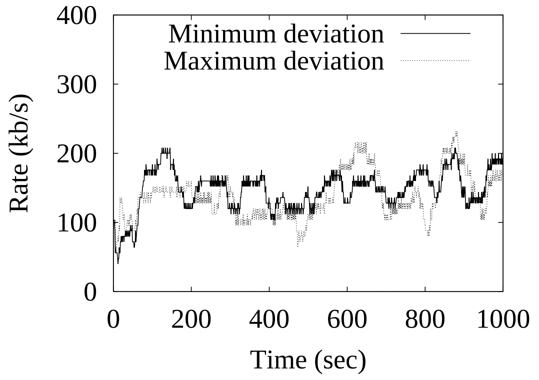
The fairness results presented in this section suggest that while CTFAF maintains the TCP-friendliness of LDA, it can successfully take various constraints imposed by the multimedia flow into account.



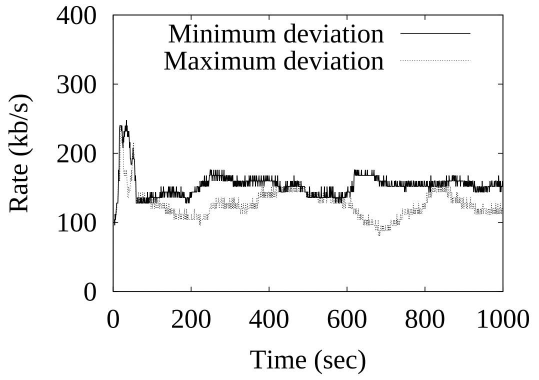
(a) $\delta_{\text{adaptation}} = \infty$ kb/s



(b) $\delta_{\text{adaptation}}$= 40 kb/s



(c) $\delta_{\text{adaptation}}$=20 kb/s



(d) $\delta_{\text{adaptation}}$= 4 kb/s

Figure 5.1: Temporal behavior of CTFAF flows with the maximum and minimum standard deviation values for different values of $\delta_{\text{adaptation}}$

5.3.2 Effects of the Length of the Resetting Period (T_{reset}) on the Performance of CTFAF

The length of the resetting period (T_{reset}) determines the interval in which B_{virtual} is reset and the bandwidth share of the flow is corrected possibly faster than the user specified adaptation rate ($\delta_{\text{adaptation}}$). Setting T_{reset} too small would mean that this correction is done rather often and would, thus, defy our goal of restricting the adaptation rate to $\delta_{\text{adaptation}}$. Using too large values of T_{reset} would on the other hand mean that the flow can not adapt rapidly enough to large changes in the network conditions.

Here, we simulated the case of competing WWW, FTP and CTFAF flows over a link of 10 Mb/s with a round trip propagation delay of 0.4 seconds and a maximum queuing delay of 0.1 seconds. The CTFAF flows used adaptation steps ($S_{\text{adaptation}}$) of 2 kb/s and the maximum possible rate change at each adaptation point ($\delta_{\text{adaptation}}$) was set to 4 kb/s. These adaptation values impose rather strict constraints on the adaptation process and a better distinction to the case of adapting the transmission rate without any restrictions ($\delta_{\text{adaptation}} = \infty$).

T_{reset} (sec)	r (kb/s)	σ (kb/s)	F	u
30	137	30	0.81	0.82
60	140	29	0.85	0.81
120	134	26	0.83	0.81
300	142	25	0.86	0.83

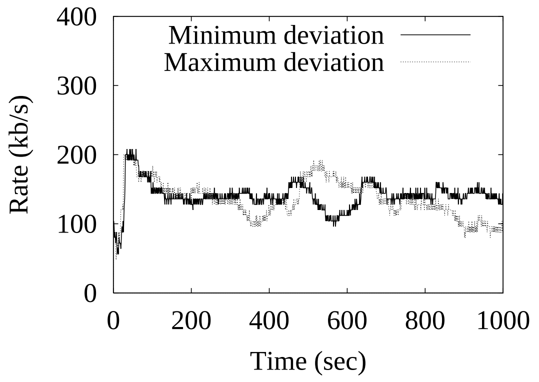
Table 5.2: Achieved average rate and fairness of CTFAF with different values of T_{reset}

The fairness and utilization results in Tab. 5.2 show little variance with varying values of T_{reset}. The average standard deviation values (σ) presented in Tab. 5.2 as well as the temporal behavior of the flows with the best and worst deviation values in Fig. 5.2 suggest that with increased values of T_{reset} the flows display a lower degree of oscillations. However, with large values of T_{reset} the flows can not react to large and sudden changes in the network conditions such as when several new flows start sending data on the same link as the CTFAF flows. Hence, setting T_{reset} to 60 or 120 seconds presents a better compromise between stability and fast reaction than a value of 300 seconds for example.

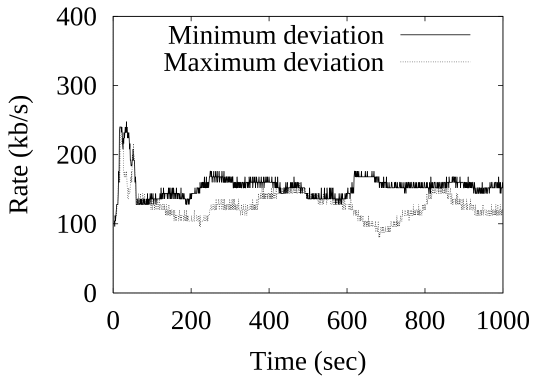
5.3.3 Effects of Delays and Flow Numbers on the Performance of CTFAF

In this part, we investigated the performance of CTFAF in terms of bandwidth utilization (u), average bandwidth share of the single CTFAF flows (r) and its fairness towards competing TCP connections (F) under different test settings with varying numbers of competing flows and delays. As a simulation topology we used the one described in Fig. 2.5 with a varying number of flows competing for a bottleneck of 10 Mb/s. In addition to n FTP and m CTFAF flows, 27 WWW servers continuously generated short TCP connections.

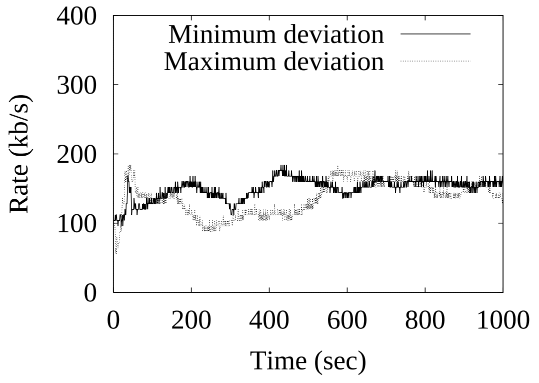
Tab. 5.3 presents the simulation results for the case of varying numbers of competing flows ($m = n = N$) and round trip propagation delays (τ). From Tab. 5.3 we can observe



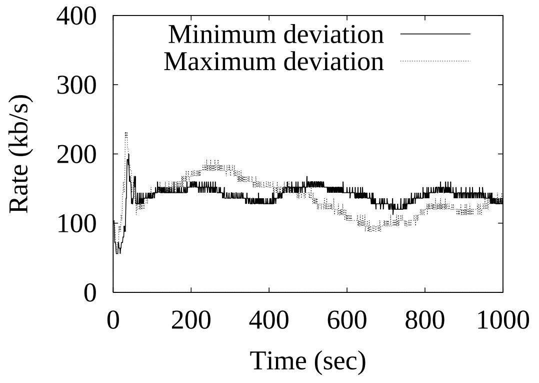
(a) $T_{\text{reset}} = 30$ sec



(b) $T_{\text{reset}} = 60$ sec



(c) $T_{\text{reset}} = 120$ sec



(d) $T_{\text{reset}} = 300$ sec

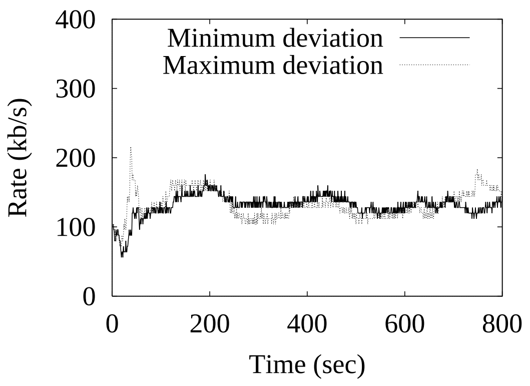
Figure 5.2: Temporal behavior of CTFAF flows with the maximum and minimum of σ for different values of T_{reset}

that the average standard deviation of the flows (σ) is less than 20% of the average bandwidth of the flows. Additionally, the deviation ($\overline{\sigma}$) in the average rate values of the single flows from the average bandwidth share of all the flows is rather negligible indicating a high degree of inter-flow fairness with all the flows receiving similar bandwidth shares. Except for the case of large propagation delays of one second, the fairness index (F) is in an acceptable range between 0.8 and 1.15 indicating similar bandwidth shares for the TCP and CTFAF flows.

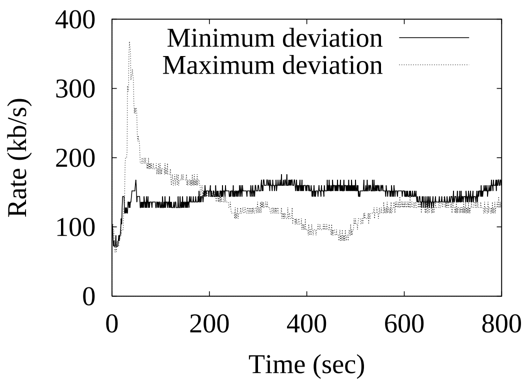
N	27					54					81				
τ (sec)	r	σ	$\overline{\sigma}$	F	u	r	σ	$\overline{\sigma}$	F	u	r	σ	$\overline{\sigma}$	F	u
0.1	160	33	7.8	0.93	0.87	99	22	5.5	1.17	0.91	75	19	4	1.3	0.94
0.2	130	27	7.5	0.71	0.83	82	20	3.4	0.87	0.9	62	16.1	4.1	1	0.92
0.4	140	29	13	0.84	0.80	76	18	3.5	0.79	0.90	55	15	2.1	0.86	0.93
0.6	169	32	16	1.14	0.85	80	20	7	0.91	0.89	54	13.9	5	0.83	0.92
1.0	234	35	17	2.47	0.89	105	22	10	1.6	0.9	65	18	6.2	1.2	0.92

Table 5.3: Achieved average rate (r) in kb/s, deviation ($\sigma, \overline{\sigma}$) in kb/s, utilization ($u$) and fairness of CTFAF with N TCP and N CTFAF competing flows, R set to 10 Mb/s and τ_q set to 0.1 seconds

Fig. 5.3 describes the bandwidth share obtained by the CTFAF flows with the maximum and minimum standard deviation (σ). Comparing the behavior of the CTFAF flows to the LDA flows under the same conditions as depicted in Fig. 4.21 we can observe that using CTFAF shows a considerable improvement in terms of flow stability as requested by the small value of $\delta_{\text{adaptation}}$ while maintaining similar performance in terms of TCP-friendly bandwidth distribution.



(a) $N = 27, \tau = 0.2, \tau_q = 0.1$



(b) $N = 27, \tau = 0.4, \tau_q = 0.1$

Figure 5.3: Temporal behavior of CTFAF

5.4 Summary

In this chapter, we presented a framework called CTFAF that integrates the congestion control process with possible constraints imposed by the compression algorithm or the communication scenario and at the same time maintains a TCP-friendly bandwidth distribution. For characterizing multimedia sources a generic model was described in Sec. 5.1 that encompasses various aspects of multimedia communication. In Sec. 5.2, a framework for integrating such a model in the congestion control process is presented. Simulating the case of deploying LDA for controlling such a generic model under the framework of CTFAF resulted in reduced oscillations and increased stability of the bandwidth shares of the competing flows and hence improved perceived QoS. To further assess the actual benefits of CTFAF the generic multimedia source model needs to be further enhanced to describe real compression styles and user preferences and objective tests need to be conducted to judge the improvements in the perceived QoS of the multimedia content.

Chapter 6

TCP-Friendly Congestion Control for Multicast Communication

As described in chapter 4, adjusting the behavior of a point-to-point data flow in accordance with the network congestion state in a TCP-friendly manner can be realized by adjusting the transmission rate of the sender in accordance with the loss and delay values measured at the receiver.

Achieving TCP-friendly rate adaptation for the case of point-to-multipoint (multicast) communication is by far more complicated. For the case of n receivers the sender would need to consider possibly contradicting values of losses and delays measured on the paths towards the different receivers. Actually, one might determine a set of $[r_1, r_2, \cdots, r_n]$ rates the sender needs to adjust its transmission rate simultaneously to, to satisfy the needs of all receivers.

For any congestion control mechanism for multicast communication to be efficient and applicable under a varying range of conditions it needs to consider the following aspects:

- **Rate adaptation:** Similar to the case of unicast congestion control, multicast congestion control should be TCP-friendly. Here, TCP-friendliness indicates that the bandwidth share consumed by the multicast flow on any traversed path resembles the bandwidth share of a TCP connection traversing the same path.

  As the bandwidth share of a TCP connection depends on its round trip delay and losses, calculating a TCP-friendly bandwidth share involves determining losses and round trip times on all paths traversed by the multicast session. Hence, multicast congestion control schemes need to support scalable and accurate loss and delay measurement approaches.

- **Scalability:** The performance of the control scheme should not deteriorate with increasing numbers of receivers. Additionally, the amount of data gathered at the end systems or transmitted between the end systems should be sustainable within the available resources.

- **Heterogeneity:** Internet links as well as the end systems connected to the Internet vary greatly in their capabilities and resources. Multicast congestion control schemes need to take this heterogeneity into account and aim at satisfying the requirements of a large part if not all possible receivers. So, for the case of n receivers one might actually determine a set of $[r_1, r_2, \cdots, r_\mathrm{n}]$ rates the sender needs to adjust its transmission rate simultaneously to, to satisfy the needs of all receivers. This might be achieved by simulcasting the same content at the different rates [104] or referring to hierarchical data transmission [113, 191] by dividing a data stream into a base layer representing the transmitted data content in a basic quality and several enhancement layers that combined with the base layer improve the overall perceived quality of the data stream. Each layer is then sent on a different multicast group. As multicast routers only forward data requested by the receivers, the receivers can determine how many sessions to join and thereby adjust their QoS in respect to their own requirements and capacities. For the case of n receivers with $[r_1, r_2, \cdots, r_\mathrm{n}]$ possible rates with r_i as an increasing value, the sender might set the rate of the lower layer to r_1, the first enhancement layer to $r_2 - r_1$ and so on. Each receiver would then join up to n layers based on its capacity. Such approaches are particularly suitable for video communication using layered compression [114].

In Sec. 6.1, we first look at some of the problems that might arise when extending a unicast congestion control scheme such as LDA for realizing multicast congestion control. In Sec. 6.2, we present a general framework for achieving TCP-friendly congestion control called the multicast enhanced loss-delay based adaptation framework (MLDA). Using MLDA, multimedia senders adjust their transmission rate in accordance with the network congestion state. For taking the heterogeneity of the Internet and the end systems into account, MLDA supports layered data transmission where the shape and number of the layers is determined dynamically based on feedback information generated by the receivers. Further, we discuss a measurement approach that allows receivers in large multicast sessions to estimate the round trip delay to the sender in a scalable way. For exchanging control information between the sender and the receivers we investigate the possibility of using the real time transport protocol (RTP) and discuss the required changes in order for RTP to support a more scalable and timely flow of feedback information from the receivers to the sender. The performance of MLDA is investigated under different test topologies in Sec. 6.4. Finally, in Sec. 6.5 the efficiency of MLDA is compared to that of other congestion control schemes proposed in the literature.

6.1 Multicast Enhancement for LDA

For LDA to support multicast communication the simplest enhancement would be for the sender to collect the RTCP messages from all receivers and determine the appropriate transmission rate towards each receiver. At specific time points the sender would then set its transmission rate based on the rates determined for the different receivers.

With each received RTCP packet from a member i the sender calculates a transmission rate (r_i) it would use if member i was the only member of the multicast session. r_i is then saved into a data base. So, if no losses were indicated the sender calculates R_{ai_i} as was described in Eqn. 4.21 and r_i is then set to

$$r_i = r + R_{\mathrm{ai}_i}$$

with r as the transmission rate the sender is currently using for the entire session. In the case of a loss indication, r_i is reduced as in Eqn. 4.25 using the loss and delay values reported by receiver i.

Instead of reacting to each RTCP packet as some adaptation schemes suggest [25], we enhance LDA to activate the adaptation process, i.e, increase or decrease the transmission rate, at certain adaptation points. At each adaptation point, the sender searches the transmission rate data base for the minimum value ($r_{\min}$) and sets r to $r_{\min}$. A $r_{\min}$ smaller than the current transmission rate (r) indicates that at least one member has reported losses. In this case, R_{ai} which is used for determining the single R_{ai_i} is reinitialized to the initial value of $\dot{R}_{\text{ai}}$. If $r_{\min}$ was higher than the current transmission rate (r), then R_{ai} is set to the difference between $r_{\min}$ and r.

We have used a period of 5 seconds between two adaptation points which is also the average value between generating two RTCP packets at the same source. As the time between two RTCP packets might actually be larger than 5 seconds, choosing this fixed value has the disadvantage that the adaptation decision might be taken based on the reports of only a subset of the session members and not all of them. However, the other alternative would be to wait for the reports from all members before adapting. While this might work for small groups, for larger groups the interval between two sent RTCP packets increases and thereby the adaptation decision will be taken less frequently and, thus, be less effective.

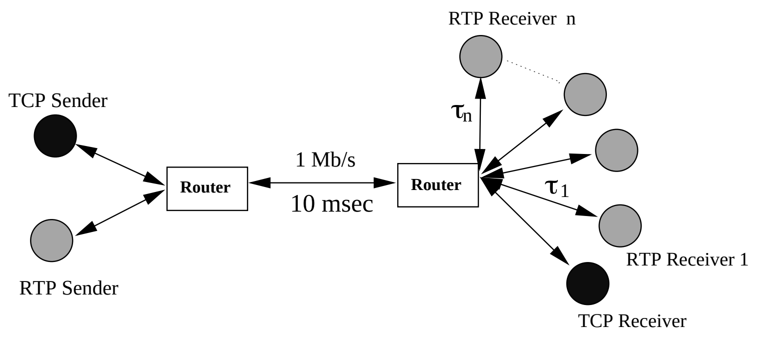


Figure 6.1: Scalability testing topology

To test the scalability of the here presented extension to LDA we used a simple topology of a congested link shared between a TCP and an LDA flow. The topology had a bottleneck bandwidth of 1 Mb/s, a round trip propagation delay of 0.1 seconds and a maximum buffering delay of 0.12 seconds. Both senders were greedy sources and always had data to send. To avoid synchronization effects and ensure equal loss probabilities for both flows random early drop was used for buffer management as was described in Sec. 2.1.1.1 with the maximum and minimum thresholds set to 0.3 and 0.8 of the available buffers. The number of multicast receivers (n) was varied between 1 and 112. The senders and receivers were connected to the bottleneck router through links of 1 Mb/s capacity and with random propagation delays between 0 and 0.01 seconds. The random delays were added to avoid synchronization among the receivers.

Fig. 6.2 depicts the bandwidth share taken by the RTP flow controlled using LDA

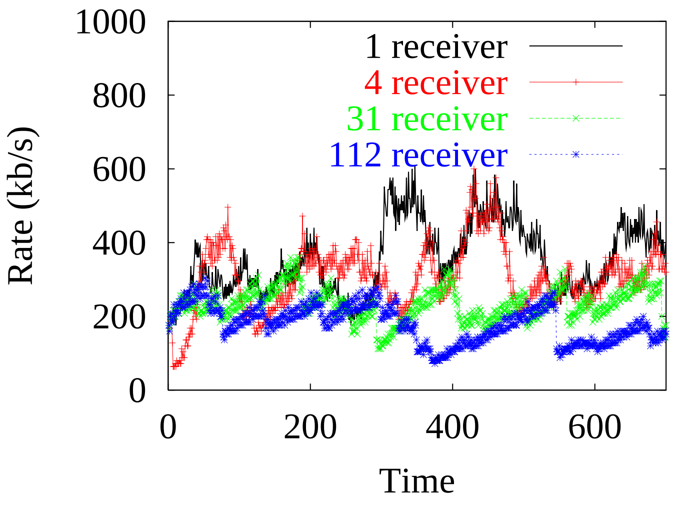


Figure 6.2: Scalability performance of multicast enhancements of LDA

enhanced for the multicast situation. While for the case of only one or four receivers the RTP flow achieves more than 300 kb/s on the average for larger numbers of receivers the average bandwidth share gets reduced below 200 kb/s.

Receivers	1	4	31	112
Bandwidth (kb/s)	348	300	212	182

Table 6.1: Bandwidth share of the RTP flow for different multicast groups with LDA

The multicast enhancement of LDA we presented here suffers from different problems. In addition to the fact that the amount of state information the sender needs to maintain increases linearly with the number of receivers, with such an approach a "race to the bottom" can occur. While we have used homogeneous receivers in the presented simulation for the case of heterogeneous receivers one poorly connected receiver would determine the quality for the much larger number of well-connected receivers.

The actual cause for the reduction in the transmission rate of the LDA sender when increasing the number of receivers relates to the RTCP interval calculation and scaling algorithm. The RTCP messages are usually sent in intervals of $(T_{\mathrm{RTCP}} \times (1 \pm 0.5))$ seconds with T_{RTCP} as the period between two successive RTCP messages sent by some receiver and is set minimally to 5 seconds. The receivers determine the loss as the average value over the entire period between sending two RTCP reports. Thus taking an adaptation interval of 5 seconds might not suffice to collect all the feedback messages from all receivers. This would then cause the sender to react to reports about the same losses different times in a row. As an example, consider the case described in Fig. 6.3. Between the first two adaptation points we measure a loss rate of 10% over a time period of 2 seconds. Assume, that the RTCP intervals of two members are 10 seconds long and are overlapping with one receiver's RTCP message arriving before the second adaptation point and the RTCP report of the other member arriving only before the third adaptation point. As the loss is averaged over the entire 10 seconds the first member would report a loss of 2%. Thereupon, the sender would decrease its transmission rate. However, the second member will still report a loss of 2% even though no losses can be measured anymore.

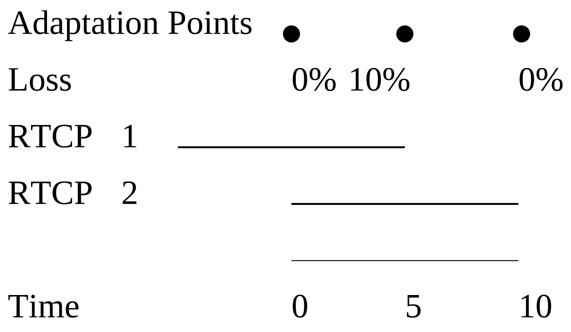


Figure 6.3: Time-line of adaptation points and RTCP report intervals

Increasing the adaptation interval might reduce the effects of this problem. However, as T_{RTCP} is increased with increasing numbers of receivers, the adaptation interval needs to be increased as well. Increasing the adaptation interval beyond a few seconds would reduce the flexibility of the adaptation interval and its effectiveness in reacting to network congestion.

The problem described here is relevant to most sender-based adaptation schemes for multicast communication that reduce the transmission rate of the sender whenever a loss indication is received and is another form of the loss multiplicity problem (LMP) [13]. LMP indicates that a sender reduces its transmission rate several times possibly down to zero as a result of independent loss indications from different receivers.

6.2 Multicast Enhanced Loss-Delay Based Adaptation Framework

To avoid having to collect state information about all receivers at the sender and avoid the loss multiplicity problem we describe in this section a general TCP-friendly congestion control approach for handling the case of heterogeneous multicast groups called multicast enhanced loss-delay based adaptation framework (MLDA). With MLDA the receivers collect information about the losses, delays and bottleneck bandwidths on the paths connecting them to the sender and determine a bandwidth share that would be utilized by a competing TCP connection traversing the same path under the same loss and delay conditions. To satisfy the needs and capabilities of heterogeneous receivers, MLDA incorporates the concept of layered data transmission in its architecture. Adaptive schemes using layered data transmission usually assume statically set layer sizes. However, to better accommodate the actual heterogeneity of the receivers, MLDA senders periodically collect information about the determined bandwidth shares at the receivers, adjust the sizes of the transmitted layers based on these information and inform the receivers about the sizes and addresses of the different layers. The receivers can then determine the number of layers to join based on the information announced by the sender and their own estimation of the TCP-friendly rate they could be using.

MLDA is a hybrid sender and receiver-based adaptation scheme that combines on the one hand various well known concepts for multicast congestion control such as receiver-based rate calculation presented by Handley et al. [61] and layered transmission [113] into a unified congestion control architecture. On the other hand, MLDA provides novel solutions for round trip delay measurements, scalable feedback collection, dynamic layering [167] and a general framework for the cooperation between the sender and the receivers that enables a seamless integration of these different concepts.

To allow for scalable feedback and yet provide the sender with enough information about the actual heterogeneity of the receivers we introduce a novel feedback approach called "partial suppression" that allows the exchange of control messages in a timely manner and to suppress the transmission of receiver information of similar content. Additionally, to get an estimation of the round trip delay between the sender and the receivers we propose a scalable measurement approach based on a combination of one-way delay estimation using timestamps of non-synchronized hosts connected by possibly asymmetrical links and end-to-end measurements of the delay.

The basic message exchange of MLDA is as follows:

1. The sender periodically transmits reports containing information about the sent layers.

2. After receiving a sender report each receiver $(j : j = 0, 1, \cdots, n)$ measures the loss and delay of the incoming data for a period of time and determines a TCP-friendly bandwidth share (r_{j}) the sender could utilize on the path connecting the sender and the receiver.

3. Based on the calculated share and the rates of the layers as reported in the sender reports the receivers decide to join a higher layer, stay at or leave the current one.

4. Further, the receivers schedule the transmission of reports indicating their calculated bandwidth share after a random period of T_{wait}. If a report form another receiver with rate indication similar to r_{j} was seen before T_{wait} expires the receiver suppresses the transmission of its report.

5. Based on the receiver reports, the sender adjusts the sizes of the different layers.

In Sec. 6.2.1 and Sec. 6.2.2 the behavior of the sender and receivers is described. Sec. 6.2.4 presents different mechanisms for estimating the loss, delay and capacity characteristics of Internet links. Note, that while the sender and receiver behavior description constitutes an integral part of the MLDA framework, the algorithms for estimating the network characteristics and the TCP-friendly bandwidth share are more generic in nature and can be easily replaced by other algorithms without altering the functionality of MLDA.

6.2.1 Sender Behavior

With MLDA the sender is responsible for adjusting its transmission behavior in accordance with the bandwidth available for the multicast session as estimated and reported by the receivers. The sender periodically polls feedback information from the receivers by generating sender reports at intervals of $(T_{\text{control}} + p)$. p is a uniformly distributed random variable that assures that the reports from senders that start at the same time do not get synchronized. The sender includes in its reports information about the number of layers (y) it is transmitting, the rate $(R_{\text{L}_k} : k = 1, 2, \cdots, y)$ of each layer (L_{k}) and the address on which each layer can be received at.

The receiver reports indicate the TCP-friendly bandwidth share estimated by the receivers to be available on the paths connecting the sender to them. The collected reports in between the sending of two sender reports are then used to adapt the sizes of the transmitted layers before sending the next sender report. To allow for fast reactions to changes in the network conditions the sender reduces the rate of the base layer whenever it receives a feedback message indicating a rate request lower than that used for the base layer.

6.2.1.1 Data Layering

Ideally, the sender would partition its data stream into a number of layers that satisfies the needs of all receivers. In a communication scenario with n receivers reporting transmission rates of $(r_1, \cdots, r_{\text{n}} : r_{\text{i-1}} < r_{\text{i}} < r_{\text{i+1}})$ the sender would then need to partition its data stream into sub-streams of $(r_2 - r_1, r_3 - r_2, \cdots, r_{\text{n}} - r_{\text{n-1}})$. Each receiver determining r_{j} as its available bandwidth share would then join x layers with $(r_{\text{j}} = \sum_{i=1}^{x} R_{\text{L}_i})$ for $(R_{\text{L}_1} = r_1 \text{ and } R_{\text{L}_i} = r_{\text{i}} - r_{\text{i-1}})$.

However, using a large number of layers might result in drift problems [71], increased delays due to the need to synchronize the different layers and higher complexity at the receivers. Therefore, as an approximation usually only a few layers are used.

An example for an approach for determining the appropriate number and sizes of the layers would be to use data mining and clustering techniques [3] that allow the sender to summarize the receiver reports into a smaller set of representative values.

Note, that any approach for dividing the data in different layers could be used instead. The layers could be chosen based on the coding used or the number of receivers requesting a certain rate if such information is available [4]. The only precondition for using such a dynamic layering approach is the availability of coders that can dynamically shape the number and sizes of layers [116].

When dividing data into layers, each layer should use its own ranges of sequence numbers. That is, while packets belonging to the same layer should have consecutive sequence numbers to allow the receivers to determine the loss rates of each layer, this is not needed among packets belonging to different layers. However, the receivers still need some means for resynchronizing the different layers into one data stream.

6.2.2 Receiver Behavior

Within the MLDA architecture, the receivers measure the characteristics of the paths connecting the sender to them, estimate the TCP-friendly bandwidth share they could consume, join the appropriate number of layers in accordance with their estimated bandwidth share and inform the sender about their estimated shares. In this section, the rules governing the join and leave actions of MLDA receivers are presented. Proposals for solving the issues of bandwidth estimation, path measurements and scalable feedback are presented in the next sections.

The join and leave actions of the receivers are triggered by the sender reports which are generated periodically in intervals of T_{control} by the sender.

After receiving a sender report the receivers measure the losses of the incoming data stream for a period of $(T_{\text{o}} \times x)$ with T_{o} as the minimum measurement time needed to obtain usable loss information. x indicates the number of layers the receiver is already tuned to. After the measurement period each receiver (j) calculates the rate (r_{j}) that would be appropriate to use on the link connecting the sender to him.

With R_{L_k} as the transmission rate on layer L_{k} the receiver can now take one of the following actions:

- $r_{\text{j}} \geq \sum_{k=0}^{x+1} R_{\text{L}_k}$: The determined TCP-friendly rate is higher than the rate of the incoming data stream in addition to the rate of the next higher layer. The receiver joins the next higher layer and starts an observation time of T_{o} to measure the loss values of the now increased incoming stream. After the measurement period a new transmission rate is determined and the receiver can again join a higher layer, leave the just joined layer or stay at the higher layer.

- $r_{\text{j}} < \sum_{k=0}^{x} R_{\text{L}_k}$: In this case, the rate of the incoming data stream is higher than the theoretically TCP-friendly share determined by some adaptation algorithm. The receiver must, thus, leave the highest layers it is currently receiving until the inequality $(r_{\text{j}} \geq \sum_{k=0}^{x} R_{\text{L}_k})$ is satisfied.

- $\sum_{k=0}^{x} R_{\text{L}_k} < r_{\text{j}} < \sum_{k=0}^{x+1} R_{\text{L}_k}$: The receiver stays at the current level.

Finally, the receiver schedules the transmission of a report carrying the value of r_{j}.

The time passing between issuing a join request for a multicast layer and receiving the data for that layer is usually only the time needed for extending the multicast distribution tree towards the receiver. Hence, joining a multicast session can be thought of as increasing the bandwidth consumed by the receiver instantaneously through the addition of a new layer. However, leaving a session results in a reduction in the rate as measured at a receiver only after a period of time. This stems from the way multicast routing entries are deleted at the routers. After receiving a leave request, the network routers that forward the multicast data to the receiver that has issued the leave request need to make sure that the session is no longer requested by any other receiver, which in general means waiting a while before stopping forwarding the data on the links towards that receiver.

MLDA receivers directly determine the number of layers they can listen to. Hence, there is no need for observing the network situation after dropping a layer and taking the leave latency into account. Note also, that losses caused by a failed join operation by a receiver do not have any effects on the loss estimation of other receivers. That is, if some receivers are listening to x layers and share a part of the multicast tree they would have an observation period of $(T_{\mathrm{o}} \times x)$. Only after the period of $(T_{\mathrm{o}} \times x)$ would the receivers be allowed to join a higher layer. In case one of the receivers tried to join the next higher layer $(x+1)$ and failed, the losses caused by the failed operation would not be included in the loss estimations of the other receivers who would have already completed their observation period.

6.2.3 Scalable Feedback

In order to take the heterogeneity of the network and receivers into account in its adaptation decision the sender needs to collect feedback information representing all receivers.

Therefore, we propose a mechanism we call *partial suppression* with which all possible rates a receiver can calculate are divided into S intervals. That is, if the possible rates a receiver could calculate might vary between R_{min} and R_{max} we divide this range into subintervals $[R_{\mathrm{min}}, R_1), [R_1, R_2), \cdots, [R_{\mathrm{S\text{-}1}}, R_{\mathrm{max}})$ with R_{min} and R_{max} as pre-defined constants. After finishing the observation period each receiver (j) calculates a theoretically TCP-friendly rate (r_{j}) and determines in which subinterval this rate is found in. The receiver schedules now the transmission of the receiver report after some time period (T_{wait}). If a report form another receiver indicating a rate in the same subinterval was seen during this period the receiver suppresses the transmission of its own report.

For realizing efficient suppression, Nonnenmacher et al. [124] suggest using a truncated exponentially distributed timer in the interval $[0, T_{\mathrm{rand}}]$ with the density of

$$T_{\mathrm{wait}} = \begin{cases} \frac{1}{\exp^{\lambda} - 1} \times \frac{\lambda}{T_{\mathrm{rand}}} \exp^{\frac{\lambda}{T_{\mathrm{rand}}} z} & 0 \le z < T_{\mathrm{rand}} \\ 0 & \text{otherwise} \end{cases} \tag{6.1}$$

For $(\lambda = 10)$ and $(T_{\mathrm{rand}} = 5c)$ with c as the delay between two receivers [124] shows analytically that for 10000 receivers less then 10 feedback messages are generated for each event the receivers are reporting on.

For dividing the possible rates into S subintervals we suggest using the following equation:

$$R_1 = R_{\mathrm{min}} \times (1+\rho) \tag{6.2}$$

$$R_{\mathrm{s}} = R_{\mathrm{s\text{-}1}} \times (1+\rho) \tag{6.3}$$

with ρ as the difference in percentage between two subsequent subintervals and $(s = 1, 2, \cdots, S)$.

The total number of subintervals (S) can then be determined as follows:

$$R_{\mathrm{max}} = R_{\mathrm{min}} \times (1+\rho)^S \tag{6.4}$$

For a possible rate range from 1 kb/s to 100 Mb/s, $(\rho = 0.1)$ and the additional restriction that the difference between two subsequent subintervals should at least be 5 kb/s the number of subintervals is 90. For the case of partial suppression we would then expect around $(S \times 10)$ receiver reports as a reaction to each sender report. Thus, if for each subinterval $[R_{\mathrm{s}}, R_{\mathrm{s+1}})$ there were a few thousand receivers that determine a rate $(r : R_{\mathrm{s}} \leq r < R_{\mathrm{s+1}})$ we would theoretically have around 900 feedback messages every T_{control} for $(\lambda = 10)$ and $(T_{\mathrm{rand}} = 5c)$.

Finally, while the sender reports are important for all receivers, the receiver reports are only of meaning to the sender and receivers of similar capacities, i.e., receivers that determine similar theoretical rates. Hence, the sender reports are sent on the base layer only. The receivers, however, should send their reports only on the highest layer they are currently listening to. This avoids overloading receivers listening to the lower layers with the reports of the receivers listening also to higher layers.

6.2.4 Receiver-Based Measurement of Path Characteristics

From Eqn. 2.6 we can see that for determining a TCP-friendly bandwidth share we need to take losses as well as delays on the links between the sender and the receivers into account. Additionally, the receiver should never ask for a bandwidth share higher than the bottleneck rate, i.e., the capacity of the smallest router, on the path connecting the sender to the receiver.

In this part of the work, we present how the loss can be determined for the case of layered transmission as well as a novel approach for measuring the round trip delay.

6.2.4.1 Loss Estimation

By requesting that data packets carry sequence numbers the loss of packets can be recognized at the receivers by gaps in those numbers. Hence, the receivers can estimate the loss as the percentage of lost data packets to the number of packets sent by the sender during an observation period. The number of sent packets can be approximated as the difference of the highest and lowest sequence numbers seen during the observation period.

To estimate an overall loss value (l) over all received layers the receiver measures the losses over each layer. For the case of listening to x layers l is determined as follows:

$$l = \frac{l_{\mathrm{L}_1} \times R_{\mathrm{L}_1} + \cdots + l_{\mathrm{L}_x} \times R_{\mathrm{L}_x}}{\sum_{k=1}^{x} R_{\mathrm{L}_k}} \tag{6.5}$$

with l_{L_k} as the loss measured over layer k and R_{L_k} as the rate of layer k as indicated in the latest sender report.

6.2.4.2 Bottleneck Bandwidth Measurement

To estimate the maximum possible bandwidth share a flow can utilize one can refer to the packet-pair approach as was described in Sec. 2.2.3.

6.2.4.3 Round Trip Delay Estimation

The simplest approach for estimating the round trip delay between two systems is by sending a request message from one system to the other and having the other system acknowledging the request right away. The round trip delay can then be estimated as the time difference between sending the request and receiving the acknowledgment for it. While this end-to-end approach works fine for unicast communication, it does not scale for the case of multicast communication. For the case of multicast with a large number of receivers, having each receiver sending a request would overload the sender and the acknowledgments would also increase the load in the network. We therefore present in this section a novel approach that combines end-to-end measurements with one-way measurements.

With this approach the sender transmits in periods of T_{control} control messages containing timestamps (T_{sender}) indicating when these messages were sent. To avoid overloading the network and the sender with receiver control messages, the receivers might use suppression or some other approach for reducing the number of control messages. Hence, each receiver sends on the average a control message every ($T_\tau : T_{\text{control}} \ll T_\tau$). The combination of receiver control messages and sender messages can then be used by the receivers to accurately estimate the round trip delay. In addition to this end-to-end measurement approach the receivers can update their estimation of the round trip delay (τ) every T_{control} using the sender's timestamps (T_{sender}). With this one-way measurement approach the receivers estimate the round trip delay to the sender as twice the difference between the timestamp of an incoming sender report (T_{sender}) and the time the report arrived at the receiver as indicated by the receiver's local clock (T_{receiver}).

$$\frac{\tau}{2} = T_{\text{receiver}} - T_{\text{sender}} \tag{6.6}$$

A smoothed round trip delay (t_{RTT}) can then be estimated similar to the approach used with TCP [179]

$$t_{\text{RTT}} = \quad t_{\text{RTT}} + 0.125 \times (\tau - t_{\text{RTT}}) \tag{6.7}$$

However, for this one-way measurement approach to deliver accurate delay estimations some kind of synchronization of the clocks among the receivers and the sender is required. Additionally, Paxon [137] shows through extensive measurements in the Internet, that the delay between two points might differ greatly in both directions. Thus, Eqn.6.6 should be reformulated as

$$\frac{\tau}{2} + \delta = T_{\text{receiver}} - T_{\text{sender}} + \sigma \tag{6.8}$$

with δ as the difference between $\frac{\tau}{2}$ and the actual one-way delay between the sender and the receiver and is in the range of $(-\frac{\tau}{2} < \delta < \frac{\tau}{2})$. σ is the offset between the clocks of the sender and the receiver due to their asynchronous behavior.

For the end-to-end measurements the receivers include in their control messages timestamps ($\dot{T}_{\text{receiver}}$) indicating when those messages were sent and the sender includes in its reports its timestamp (T_{sender}). Additionally, the sender reports include for each seen receiver message an entry indicating the identity of the reporting receiver, the time passed in between receiving the report and sending the sender report (T_{DLSR}) as well as the timestamp included in the receiver report ($\dot{T}_{\text{receiver}}$). The receivers indicated in the receiver list of the sender message can get an accurate value of τ and an estimation ($\theta : \theta = \delta - \sigma$) of the link asymmetry and of the offset between the clocks. θ can then be determined from the following equations:

$$\tau = T_{\text{receiver}} - \dot{T}_{\text{receiver}} - T_{\text{DLSR}} \tag{6.9}$$

$$\theta = T_{\text{receiver}} - T_{\text{sender}} - \frac{\tau}{2} \tag{6.10}$$

θ is then updated with each end-to-end measurement. In between two end-to-end measurements, the receiver updates its estimation of τ using θ and the timestamps of the periodically arriving sender reports as in Eqn. 6.8.

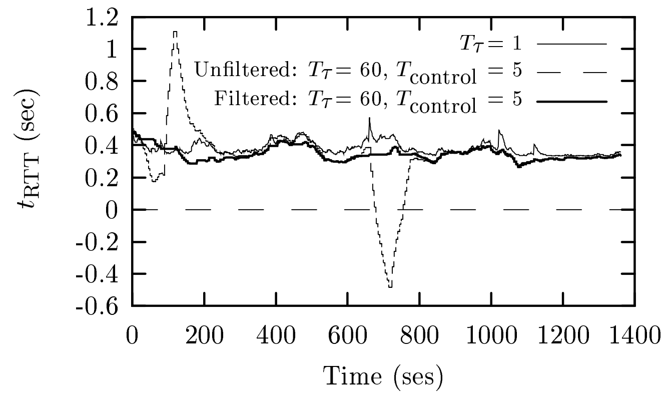


Figure 6.4: Delay measurement with MLDA

We have tested the accuracy of this delay measurement approach by running several measurements between Berlin and Berkeley. The results depicted in Fig. 6.2.4.3 show the estimated smoothed round trip delay (t_{RTT}) when using an end-to-end round trip delay measurement every one second ($T_\tau = 1$) and for the case of running an end-to-end measurement every 60 seconds ($T_\tau = 60$) and updating t_{RTT} using one-way measurements every 5 seconds ($T_{\text{control}} = 5$). The initial round trip delay was set to 0.5 seconds. In general using our estimation mechanism resulted in values very close to those determined by running an end-to-end measurement every second. However, in Fig. 6.2.4.3 we could identify three sources of problems:

1. As the system clocks of the different hosts run at different speeds, a skew in the one-way delay measurement exists that leads to an error in the calculation.

2. Due to some irregularities in the network or the end systems the end-to-end measurement might be wrong which would lead to wrong estimates of θ followed by wrong estimates of t_{RTT}. For example, the measurement done at time 660 seconds resulted in a high round trip delay. As this value was used for updating θ this lead to negative one-way delay estimations and negative values of t_{RTT}.

3. Due to their inaccuracy, system clocks of computers are usually adjusted using a more accurate clock that is otherwise not directly accessible by the user[1]. This

[1]This adjustment occurs on the tested Solaris systems, when the system clock differs by more than 1 second from the more accurate clock.

resulted in large jumps in the order of one second in the measured one-way delay and in false τ values. This effect can be seen in the sudden increase in t_{RTT} at the 120th. second.

To filter out such irregularities we used the following approach:

- **End-to-end measurement:** For the case that an end-to-end measurement resulted in a τ with

  $$|\tau - t_{\text{RTT}}| > t_{\text{RTT}} \times \xi \tag{6.11}$$

  then we save the value of the last calculated θ to θ_{old} and τ to τ_{old} and determine a new θ as in Eqn. 6.10.

  If using the newly calculated θ with the next N one-way delay measurements delivers estimations of τ that are in the range of $[\tau_{\text{old}} \times (1 \pm \gamma)]$ then the newly calculated value of θ is used until the next end-to-end measurement. Otherwise, the receiver uses θ_{old} until the next measurement after which a new θ is determined.

- **One-way measurement:** For each one-way measurement a value ϵ is calculated as

  $$\epsilon = T_{\text{receiver}} - T_{\text{sender}} \tag{6.12}$$

  For the case that the determined τ after a one-way delay measurement fulfills the following inequality

  $$|\tau - t_{\text{RTT}}| > t_{\text{RTT}} \times \xi \tag{6.13}$$

  we save the newly determined τ to τ_{tmp} and determine a value ϕ as the difference between the value of ϵ calculated during the last one-way delay measurement that did not fulfill Eqn. 6.13 and the ϵ determined for the first measurement that fulfilled Eqn. 6.13.

  If the next N one-way delay measurements deliver estimations of τ in the range of $[\tau_{\text{tmp}} \times (1 \pm \gamma)]$ then θ is increased by ϕ otherwise the measurement that has delivered τ_{tmp} is assumed to be wrong and is ignored.

Applying these filtering rules to the measurement results in Fig. 6.2.4.3 with N set to 3, ξ set to 0.5 and γ set to 0.5 leads to a smoothed round trip estimation with an error of maximally 20% from the one achieved with an end-to-end measurement every second.

Finally, note that the skew of the clocks plays a minor role here as its effects are offset by the end-to-end measurements. However, when increasing the interval between two end-to-end measurements, i.e., using larger values of T_τ, an approach for estimating the skew such as in [118] should additionally be used.

Having each receiver sending a report every T_τ would result in scalability problems even for the case of large values of T_τ on the order of a few minutes. On the other hand, using a suppression mechanism might cause some receivers to never send a report and thus not be able to do an end-to-end measurement. Therefore we extend the scheme with the notion of local representatives. That is, receivers that send a report and make an end-to-end measurement announce to their neighboring receivers the determined θ. This is done by sending a special packet to the basic multicast layer with the time to live set to a small value to ensure that the packet will only reach geographically close members of the session. While we can assume that all neighboring receivers suffer from the same asymmetry to the sender, their clock offsets from the sender still differ. Thus, in the announced θ only the δ term is valid for the neighboring receivers. To get an estimation of the synchronization term (σ) of each receiver to the sender, the receivers start an end-to-end measurement of the round trip delay and a local θ (θ_{local}) to the announcing receiver. This is done similarly to the end-to-end measurement to the sender and calculation in Eqn. 6.10 but with the announcing receiver taking the role of the sender. Adding θ_{local} to the θ announced by that receiver all receivers get a valid estimation of their own θ with the sender.

6.2.5 Synchronous Join and Leave Actions

For the case of multicast, a data stream traverses a network node as long as at least one receiver behind this node is listening to this layer. Thus, the action of leaving a layer results in an overall rate reduction at this node only if all receivers behind this node leave this layer as well. Additionally, if two receivers joined different layers at the same time, the receiver joining the lower layer might observe losses that were not caused by the rate increase due to his join action but by the other receiver joining a higher layer.

To coordinate the actions of the different receivers we use the end of the observation periods as implicit synchronization points. That is, receivers can try to join the first enhancement layer after the end of the observation period of the basic layer (T_{o_0}). Receivers listening also to the first enhancement layer can join the second one or leave the first enhancement layer at the end of the observation period for the first enhancement layer, i.e., ($T_{o_0} + T_{o_1}$) after receiving the sender report.

Using synchronization points in itself is not novel. Vicisano et al. [191] already use this approach based on specially flagged packets. Note, however, that due to the heterogeneity of the network the sender reports (or the specially flagged packets) arrive at the receivers at different time points depending on the delay between the sender and the receiver. To reduce the effects of this delay variation we extend the observation period for the basic layer by ($T_{\text{sync}} = \frac{\tau_{\max} - t_{\text{RTT}_i}}{2}$) with $\tau_{\max}$ as the maximum seen round trip delay between the sender and the receivers in the multicast session and t_{RTT_i} as the estimated round trip delay at receiver i. For estimating $\tau_{\max}$ the receivers include their estimation of the round trip delay (t_{RTT_i}) to the sender in the receiver reports. The sender determines then based on the receiver reports the maximum round trip delay ($\tau_{\max}$) and includes this value in its reports. Note that as not all receivers send reports, the $\tau_{\max}$ estimation at the sender might not be completely correct. Additionally, as the t_{RTT_i} of the receivers are only estimations and the round trip delay tends to vary on short time scales this approach does not guarantee a perfect synchronization among the receivers but still manages to improve it compared to simply relying on the sender indications.

6.2.6 Reliability Issues

The dependency on the sender reports for initiating the adaptation actions of the receivers might lead to a deadlock situation. During overload periods the sender reports might get lost. However, without these reports the receivers do not start the observation periods and then leave the higher layers in case of losses. Hence, in this case the congestion situation prevails and more sender reports might get lost. To avoid this situation, the receivers schedule a new observation period after a timeout of $(\sigma \times T_{\text{control}} : \sigma > 1)$ with T_{control} as the time period between sending two reports at the sender. Thus, if the sender report was lost, the receiver would start an observation period maximally $(T_{\text{control}} \times \sigma)$ seconds after receiving the last sender report. After the observation period, the receiver can join or leave a layer and schedule a receiver report as was described above. In case a sender report was received before the timeout expires the scheduled actions are canceled and new ones are scheduled.

6.2.7 Parameter Settings

In the description of MLDA we have used several parameters that control the temporal behavior of the algorithm. The sender transmits a report every $(T_{\text{control}} + p)$, the receivers measure the behavior of a layer for T_{o} and schedule the transmission of a report in a period of $[0, T_{\text{rand}}]$.

Further, we need to consider observation periods for different layers and accommodate possible delays in the leave actions. We therefore set T_{control} to 10 seconds and p arbitrarily to 0.5 seconds. To allow for a good degree of suppression we set T_{rand} to 1.5 seconds which is 5 times a typical half of the round trip delay between Europe and the States as measured between Berlin and Berkeley. Finally, note that for taking the adaptation decision based on the rates determined by all receivers, the receiver reports triggered by a sender report need to arrive at the sender before sending the next report. Therefore, the time left for the observation periods for all layers $(T_{\text{o}_{all}})$ can be determined as:

$$T_{\text{o}_{all}} = T_{\text{control}} - p - T_{\text{rand}} - T_{\text{return}} - T_{\text{sync}} \tag{6.14}$$

with T_{return} as the time it takes the receiver reports to arrive at the sender and should be set at least to half the maximum round trip delay. With T_{return} and T_{sync} set arbitrarily to 0.5 seconds $T_{\text{o}_{all}}$ has a value of 7 seconds. After several simulations we decided to use an observation period of at least 1.5 seconds as using smaller values resulted in inaccurate loss values and a highly oscillative adaptation behavior. This means that the scheme supports only up to around 5 layers. While this naturally presents a limitation, using a larger number of layers would, however, increase the complexity of the receivers as they need to resynchronize the data from different layers. As the different layers might actually take different routes to the receiver, they might suffer from different delays. To reconstruct a data segment consisting of data packets sent over different layers the receiver would need to wait until all the different parts are received. This incurs additional delay and thus might reduce the overall perceived quality.

6.3 Integration of RTP and MLDA

To allow for the cooperation of the sender and receivers in the MLDA framework different control information need to be exchanged. The sender periodically transmits a report with a description of the transmitted layers and some timing information. The receivers need also to send feedback information to the sender with timing and bandwidth information. The data packets themselves need to have sequence numbers in order for the receivers to detect losses as well some information that allow the receivers to resynchronize data of different layers into one data flow.

A protocol that already supports a large part of those requirements is the real time transport protocol (RTP) [158] widely used for multimedia communication in the Internet for carrying control information between the senders and receivers. RTP [158] defines a data and a control part. For the data part RTP specifies an additional header to be added to the data stream to identify the sender and type of data. With the control part (RTCP), each member of a multicast session periodically sends control reports to all other members containing information about sent and received data. Additionally, the end systems might include in their reports an application specific part (APP) intended for experimental use. For more details see Sec. 2.1.3.1.

RTP already includes sequence numbers in the data packet headers allowing the receivers to detect losses. Additionally, RTP includes timestamps in the data packets that enable the receivers to reestablish a timely sequence of the received data from different layers and hence to resynchronize different layers into one data stream.

However, to accommodate the different needs of MLDA some further additions are required for the integration of MLDA and RTP:

Time measurements: For the time measurements, the session members already include in the RTCP reports timestamps indicating the report generation time. Additionally, each receiver indicates in its reports the timestamps of all sender reports seen since sending the last report, the identities of the senders and the time passed between receiving those reports and sending the receiver report. This allows the senders to estimate their round trip delay to the receivers.

For the case of MLDA the receivers need to estimate their round trip delay to the sender as well. Hence, in this case, the senders need to include in their reports the timestamps of the received receiver reports since sending the last sender report, the identities of the reporting receivers and the time passed between receiving each report and generating the sender report.

Bottleneck measurement: As already described in Sec. 4.3.1, in order to estimate the bottleneck bandwidth the sender needs to transmit a row of back-to-back data packets. To inform the receivers which data packets should be considered as probe packets, the RTCP sender can add to its reports an application defined part (APP) including the source sequence number, the sequence number (SEQ) of a data packet that will start a stream of probe packets and the number (n) of probe packets that will be sent. Then, n data packets starting with the packet numbered SEQ are sent at the access speed of the end system. The receiver can then use the time gaps between those packets for estimating the bottleneck.

Control information: The MLDA sender needs to inform the receivers about the sizes, addresses and number of transmitted layers as well as the maximum seen round trip delay. This information can be included in the sender RTCP reports as an application specific part. To allow the sender to dynamically change the adaptation parameters the timing information such as T_{o} and T_{rand} should be included in the sender reports as well. Similarly, the receivers can include in their RTCP reports their estimated bandwidth shares as application specific parts.

Control periods: As already mentioned in Sec. 6.2 the sender transmits its reports in fixed periods. After receiving a sender report each receiver schedules a feedback report and uses the partial suppression approach for reducing the total number of reports.

The RTCP traffic is, however, scaled with the data traffic so that it makes up a certain percentage of the data rate (usually 5%) with a minimum interval of 5 seconds between sending two RTCP messages. For large sessions this might result in very scarce reports not enough for achieving efficient adjustment of the sender behavior. Hence, this part of the RTCP specification needs to be adjusted to allow for a more scalable and timely flow of feedback information from the receivers to the sender.

If some receivers of the multicast data do not support MLDA, the APP parts can be ignored. These receivers can still join the multicast session and receive the data of the basic layer. A receiver driven adaptation scheme such as [113] can be used to join higher layers.

6.4 Performance Investigation of MLDA

In this part of the work, we study the behavior of MLDA using simulations. Here, we mainly concentrate on the TCP-friendliness of MLDA and its ability to accommodate the needs of heterogeneous receivers.

6.4.1 Simulation Model of MLDA

For the rate estimation part of MLDA we used in this study the loss-delay adaptation algorithm (LDA) [177] as described in Sec. 4.3. However, in contrast to the approach presented in Sec. 4.3 where the sender calculated the bandwidth share of the flow based on the loss and delay information from the receiver, here the receivers determined the TCP-friendly rate by themselves and informed the sender about this value.

For dividing the data into K layers we referred here to a simple approach. The sender determined the minimum (r_{min}) and maximum (r_{max}) reported rates and set the rate for the basic layer to r_{min}. The enhancement layers were then set to $(\frac{(\alpha \times r_{\text{max}}) - r_{\text{min}}}{K-1})$. ($\alpha : \alpha \leq 1$) was used as a dampening factor that reduced the effects of a possible overestimation of the available resources by the receivers. To avoid establishing layers with a negligible content, the minimum size of a layer was set to 10 kb/s. In the simulations presented here, we set α to 0.9 and K to 3. Note that while the chosen layering approach might have some impact on the fairness and stability of MLDA, the actual performance of MLDA does not depend on using a particular layering scheme.

The exchange of signaling information was realized in the simulation model using a model of RTP with the extensions described in Sec. 6.3. As a simplification, an accurate estimation of the bottleneck bandwidth was available at the receivers right after the start of the simulation. We additionally assumed that the clocks of the sender and receivers were synchronized and that the round trip delay was measured accurately. This allowed us to better investigate the aspects of TCP-friendliness and stability of the scheme without having to consider possible side effects of measurement errors.

Finally, we assumed that the time passing between sending a leave request and this leave action actually taking effect to be constant and was set to one second.

6.4.2 Performance of MLDA in Heterogeneous Multicast Environments

To test the performance of MLDA in heterogeneous multicast environments we used the topology depicted in Fig. 6.5 with a multicast session consisting of a sender and 6 receivers connected to the sender over routers with different capacities. Each router was shared between an MLDA stream and m TCP connections that had the same end-to-end propagation delay as that of the MLDA sender/receiver pair. The TCP connections were modeled as FTP flows that always had data to send and lasted for the entire simulation time. A TCP-Reno [179] model was used for simulating the congestion control behavior of TCP. The sender transmitted packets of 1 kbytes and each router was a random early drop (RED) [55] router. Here we used a maximum queuing delay of 0.15 seconds and set the maximum and minimum thresholds to 0.8 and 0.3 of the maximum buffer size. The router R0 worked only as a distributer of data and incurred no losses or delays to the data. In our simulations, we set the round trip propagation delay between the sender and the receivers to 200 msec.

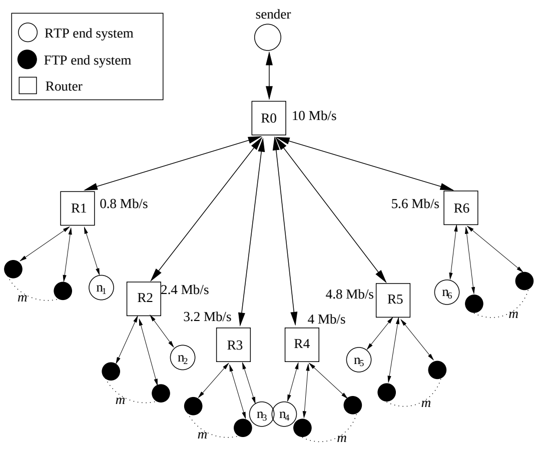
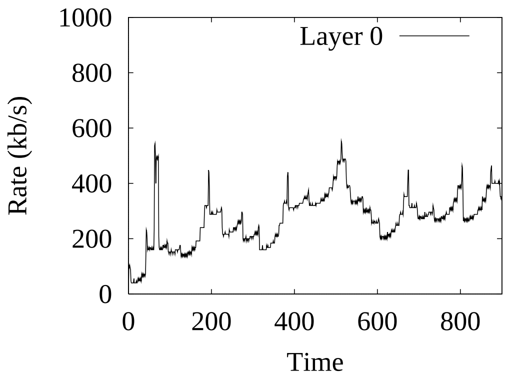
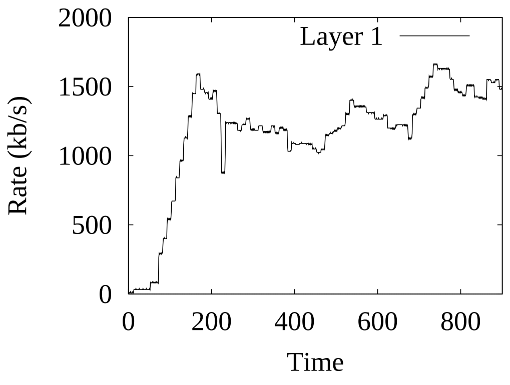


Figure 6.5: Multicast testing topology

Fig. 6.6 shows the distribution of bandwidth among the different layers for the case when each router is shared between an MLDA flow and a TCP connection. We can observe that the rate of the basic layer, see Fig. 6.6(a), is adjusted in accordance with the capacity of receiver n_1.



(a) Layer 0



(b) Layer 1, 2

Figure 6.6: Bandwidth distribution of the layers

Fig. 6.7 displays the bandwidth distribution between the competing MLDA and TCP flows at the different routers.

For the case of different numbers of competing TCP connections (m) Tab. 6.2 indicates that the ratio of the average bandwidth share of the MLDA receivers to the average bandwidth share of the competing TCP connections at each router (F_{n}) varies around one. Both the results in Fig. 6.7 and Tab. 6.2 indicate that MLDA is in general fair towards the competing TCP connections and manages to satisfy the capabilities of the heterogeneous receivers. While the bandwidth shares of the receivers with the lower bandwidth (n_1 and n_2) is restricted by the adaptation scheme, receivers that have a TCP-friendly bandwidth share that is larger than the sum of the first x layers but below the sum of the $x+1$ layers will oscillate between layer x and $x+1$. Depending on the chosen layering approach and number of layers this might result in temporary unfair bandwidth distribution. Improved fairness and smaller oscillations can only be reached using a larger number of layers [150] which, however, increases the complexity of the scheme as the end systems need to manage and resynchronize a larger number of layers.

Flows (m)	F_1	F_2	F_3	F_4	F_5	F_6
1	0.66	0.70	1.03	1.04	1.5	0.95
8	0.50	0.90	0.85	1.09	0.92	1.02

Table 6.2: Ratio of average bandwidth share of the MLDA flow to the average share of competing TCP connections

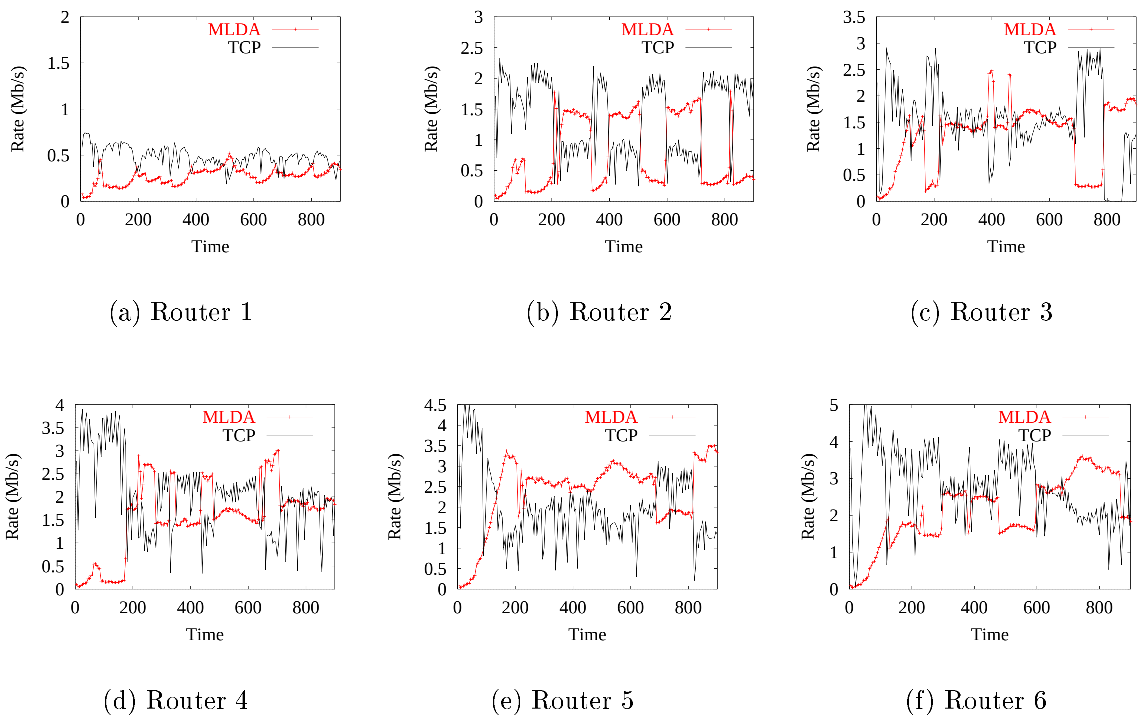


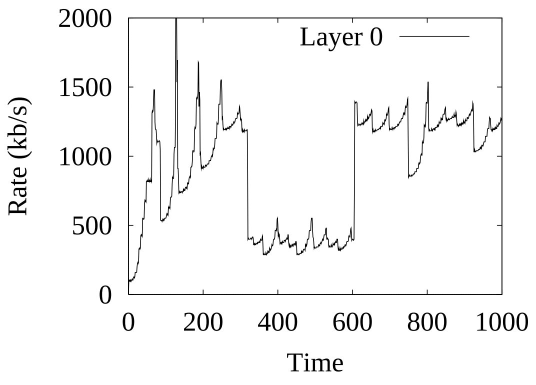
Figure 6.7: Bandwidth distribution between TCP and MLDA at the different routers

6.4.3 Performance of MLDA in Dynamical Environments

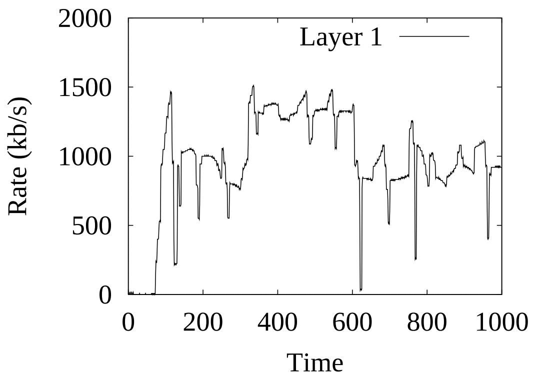
In this section, we investigated the behavior of MLDA in a dynamical receiver setting, i.e., a setting with the receivers joining and leaving the multicast session during the simulation time.

For this purpose, we used the simulation topology of Fig. 6.5 but with receiver (n_1) joining the multicast session at time 300 seconds and leaving it at time 600 seconds. Each router was shared between the MLDA flow and uncorrelated background traffic which consumed maximally half of the router's capacity. The background traffic was modeled as the aggregate of 100 on-off processes with the on period lasting for the time needed to carry a number of packets drawn from a Pareto distribution with the factor of 1.1 and a mean of 20 packets at a rate of $(\frac{R}{200})$ with R as the rate of the router traversed by this traffic. The off periods lasted for a time drawn from a Pareto distribution with a factor of 1.8 and a mean of 0.5 seconds [133].

As Fig. 6.8 depicts, during the first 300 seconds the bandwidth share of the lowest layer is increased up to around 1.2 Mb/s which is the bandwidth share of receiver n_2 constituting the worst receiver in this case. After receiver n_1 joins the session the size of the base layer is reduced down to the appropriate level of the new worst receiver, i.e., 400 kb/s. The sizes of the upper layers is increased to compensate the reduction in the base layer. Thus the receivers listening to the higher layers are not affected by the reduction in the base layer. The share of receiver (n_6) as depicted in Fig. 6.9(f) is not



(a) Layer 0



(b) Layer 1, 2

Figure 6.8: Bandwidth distribution of the layers

changed due to the join and leave actions and is comparable to the results achieved in Sec. 6.4.2, see Fig. 6.7(f). After receiver n_1 leaves the session at 600 seconds the size of the base layer is increased again and, hence, improving the bandwidth share of the receivers listening only to the lower layer.

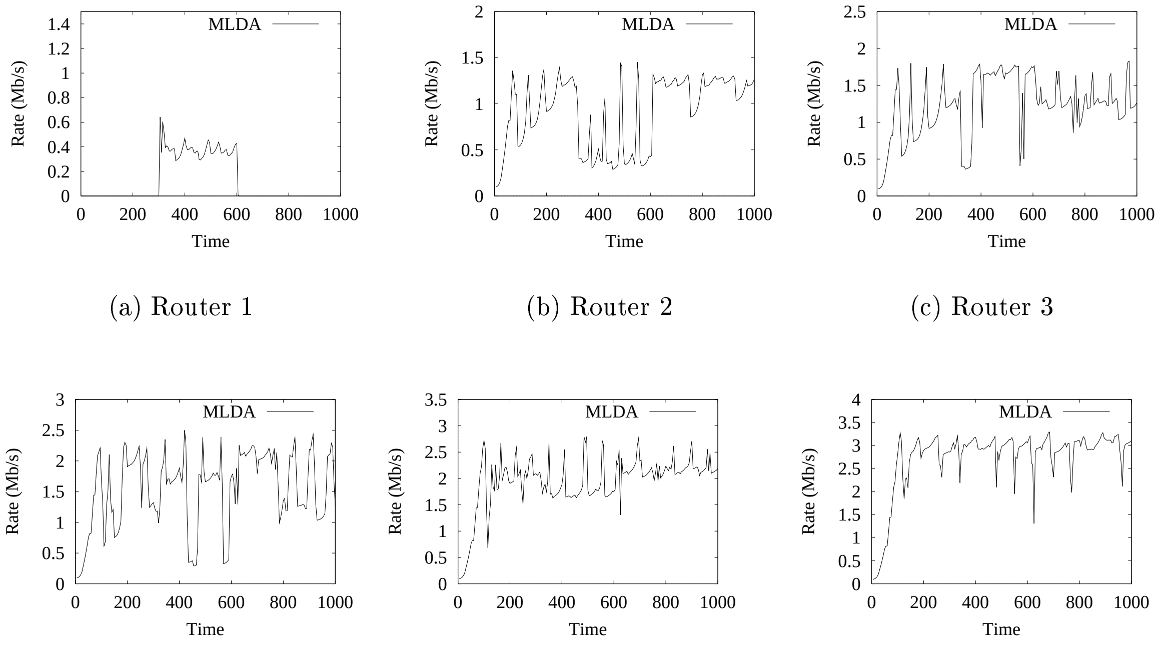
6.4.4 Performance of MLDA in Multicast Environments with Shared Links

In the previous sections, see Sec. 6.4.2 and Sec. 6.4.3, we only considered multicast distribution trees without any shared links among the receivers. This was beneficial for investigating the TCP-friendliness of MLDA and its ability to accommodate heterogeneous receivers without having to consider the effects of interactions between different receivers, traversing multiple routers and different round trip delays among the receivers.

Fig. 6.10 depicts another configuration of a multicast tree that is shared among different receivers with each link of the tree having a different capacity. Similar to Fig. 6.5, RED routers are used. Each router is shared between the MLDA flow and the uncorrelated background traffic which consumes maximally half of the router's capacity.

The bandwidth distribution among the layers, see Fig. 6.11, accommodates in this case the capacities of the receivers such that receiver n_5 manages to get a bandwidth share of around 170 kb/s and the receivers connected to the first router ($R1$) get a bandwidth share of 540 kb/s.

Fig. 6.12 suggests that receivers connected to the same router over links with similar bandwidth capacities but with different round trip delays receive identical bandwidth shares and stay synchronized throughout the simulation time. Also, notice that receiver n_4 oscillates between the first two layers which might cause losses in the stream forwarded further towards receiver n_5. However, due to the synchronization mechanisms of MLDA those losses are ignored at receiver n_5 which does not include the losses caused of the



(a) Router 1 (b) Router 2 (c) Router 3

(d) Router 4 (e) Router 5 (f) Router 6

Figure 6.9: Bandwidth share of MLDA at the different routers

failed join operations of receiver n_4 into its loss estimations.

6.4.5 Scalability Performance of MLDA

As the actual rate calculation procedure is done at the receivers, the sender needs only to collect the rate values indicated in the receiver reports and use these values for determining the number and size of layers to use. Thus, with increasing number of receivers the amount of data and processing overhead at the sender does not increase considerably. For the scalability tests we repeated the simulation done in Sec. 6.1 with an MLDA and a TCP flow sharing a bottleneck link, see Fig. 6.1. Unlike the results depicted in Fig. 6.2 the bandwidth share of the MLDA controlled flow does not change considerably with changing the number of receivers, see Tab. 6.3.

Receivers	1	4	31	112
Bandwidth (kb/s)	328	300	311	290

Table 6.3: Bandwidth share of the MLDA flow for multicast groups of different sizes

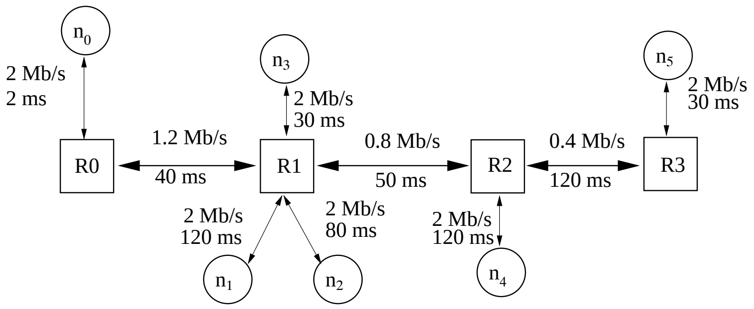
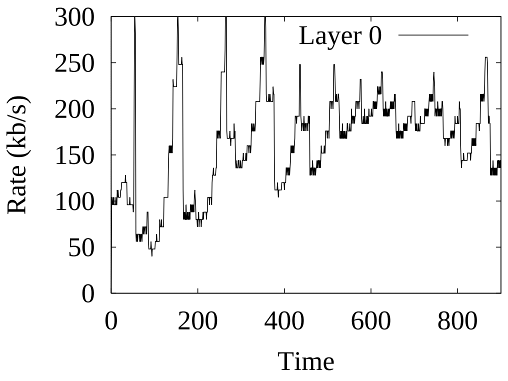
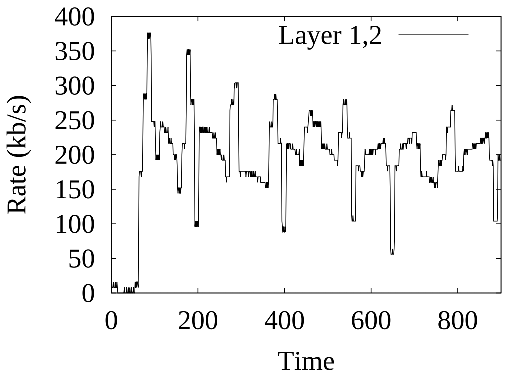


Figure 6.10: Testing topology of shared multicast trees



(a) Layer 0



(b) Layer 1,2

Figure 6.11: Bandwidth distribution of the layers in the shared tree scenario

6.5 Comparison of the Performance of MLDA to Other Congestion Control Schemes

In this part of the work, we compare between the performance of MLDA and a row of recent proposals for TCP-friendly congestion control schemes for multicast communication. Due to the large number of such proposals we restrict our comparisons to only a few that combine TCP-friendly adaptation with layered data transmission. For comparing the schemes, we picked out some representative test cases as were described in the papers presenting those algorithms. We re-simulated those cases in our simulation environment and compared the achieved results using MLDA with the results achieved by the other schemes as were reported by their authors. This approach reduces possible errors in the comparisons due to misinterpretations or wrong implementations of the algorithms.

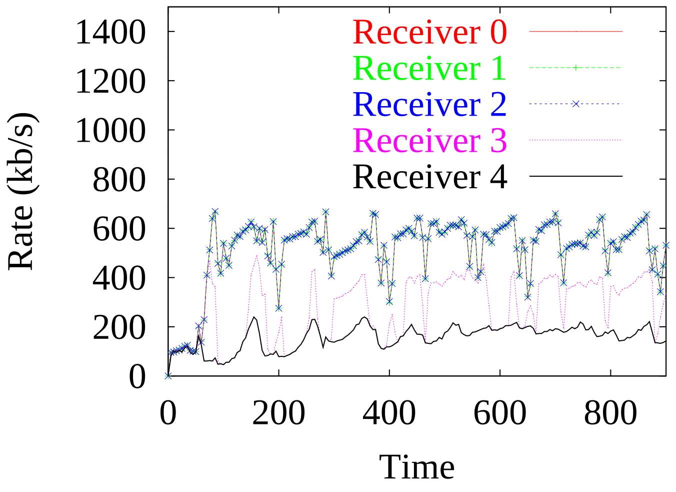


Figure 6.12: Bandwidth distribution among the receivers in the shared tree scenario

6.5.1 Comparison of MLDA and PLM

The packet-pair receiver-driven layered multicast (PLM) [102] is based on layered transmission and on the use of the packet-pair approach to infer the bandwidth available at the bottleneck to decide which are the appropriate layers to join. PLM is receiver driven, i.e., congestion control is achieved through the join and leave actions of the receivers. The packet-pair approach is used usually to estimate the bandwidth of the bottleneck router on the path between the sender and the receiver and not the bandwidth share of the transmitted flow. To apply the packet-pair approach for reliably estimating a flow's bandwidth share, the authors of PLM assume that all routers in the network deploy some kind of a fair queuing mechanism that allocates each flow a fair bandwidth share. Only under this assumption, which is currently invalid in the Internet, is it possible to use PLM for congestion control.

With PLM the sender periodically sends a pair of its data packets as a burst to infer the bandwidth share of the flow. The receivers use the gaps between the specially marked packet-pairs to estimate their bandwidth share. Based on the estimated share they determine the number of data layers they can receive.

For comparing the TCP-friendliness of PLM and MLDA we used the topology depicted in Fig. 6.14 with a bottleneck link shared between two TCP ($m = 2$) connections and an MLDA flow ($n = 1$). The bottleneck link had a bandwidth of 300 kb/s and a delay of 0.02 seconds. For reducing the effects of synchronization and ensuring a fair distribution of the losses we used a RED router with a maximum buffering delay of 0.2 seconds and the maximum and minimum thresholds set to 0.3 and 0.8. The initial transmission rate of the MLDA flow was set to 100 kb/s and the packet size was set to 500 bytes. The first TCP connection started at time 0 seconds and the second one 60 seconds later. The adaptive flow started at time 20 seconds. For PLM, a packet-pair was sent each second and each flow had a queue size of 20 packets. Additionally, the sender transmitted 17

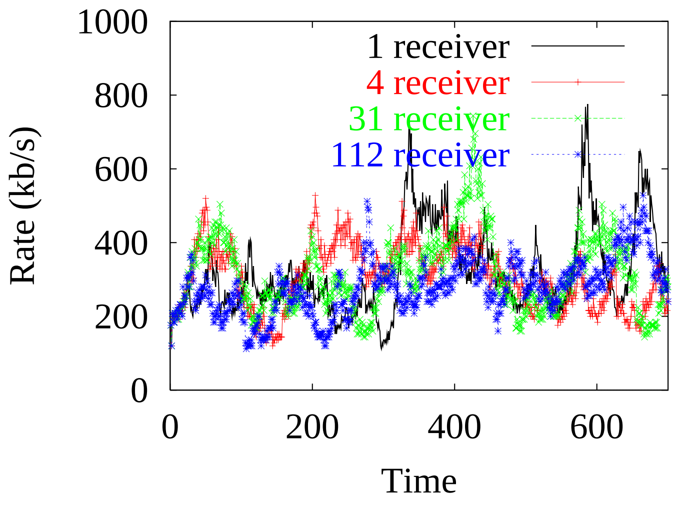


Figure 6.13: Scalability performance of MLDA

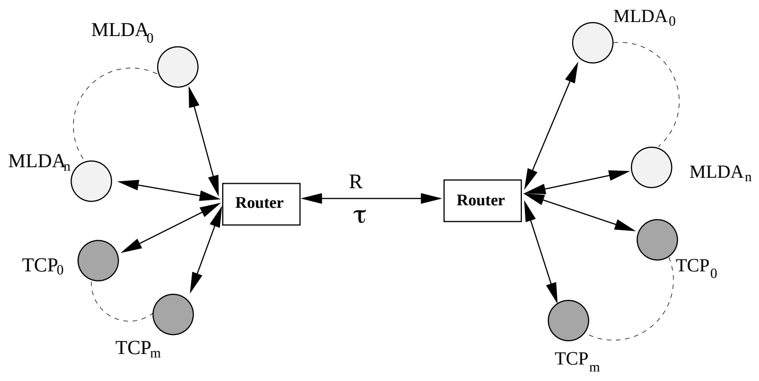
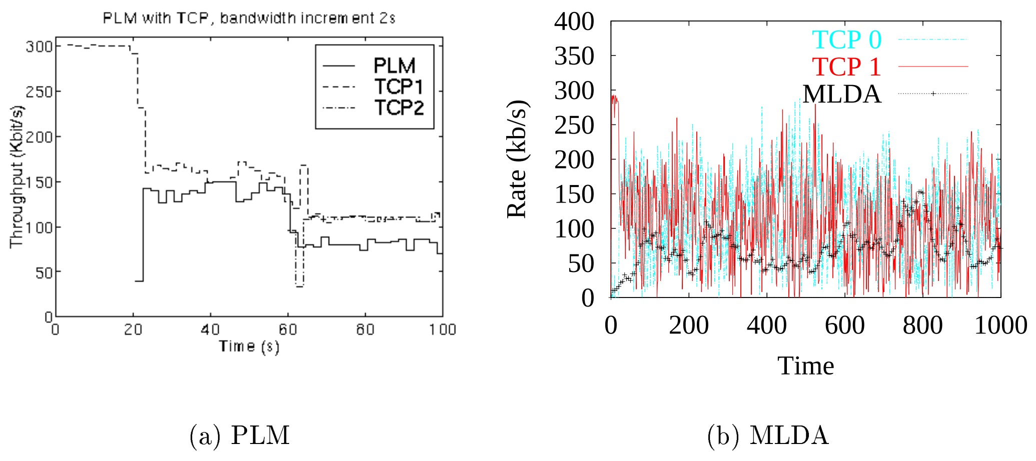


Figure 6.14: Testing topology for comparing MLDA with other schemes

layers each of 20 kb/s. This topology was taken from [102] Fig.[2] whereas in [102] a router with fair queuing was used.

The results presented in Fig. 6.15 describe the bandwidth distribution between the TCP and MLDA flows when using PLM and MLDA. Measured over the entire simulation time, both approaches achieve similar average bandwidth distributions with the adaptive flow receiving around 80 kb/s and the TCP flows having a share of 110 kb/s each. However, looking at the temporal behavior of the flows in Fig. 6.15 it can be observed that with PLM the flows show hardly any oscillations and have rather constant bandwidth shares. With MLDA on the contrary, the TCP flows oscillate between 0 kb/s and 250 kb/s. While the MLDA flow itself shows a less oscillatory behavior than TCP it still shows a variance of ±50% of its average bandwidth share. With PLM the receivers always know their exact bandwidth share and as the number of flows in the network is constant the bandwidth shares of the flows is constant as well. Hence, this stable behavior of the flows with PLM was to be expected.



(a) PLM (b) MLDA

Figure 6.15: Bandwidth distribution with MLDA and PLM

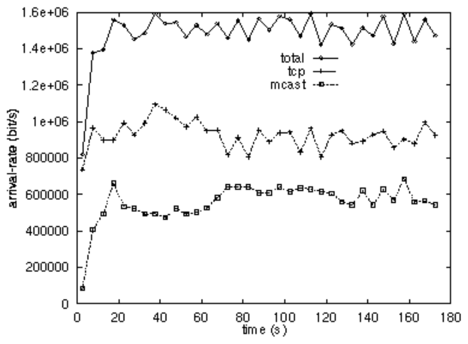
Note that PLM not only assumes a fair queuing network but also that the packet-pair approach always results in correct estimations of the bandwidth share. However, even in a fair queuing network, the packet-pair approach might result in under or overestimation of the bandwidth share. That is, due to losses of probe packets, network congestion or interference of other traffic this approach might result in wrong estimations of the bandwidth. Hence, the receivers need to deploy mechanisms such as [97, 137, 29] to filter out wrong estimates. As such effects are difficult to simulate, the results presented in [102] for PLM are to be considered rather optimistic. Due to the effects of losses and traffic interference a higher degree of oscillations can be expected in a more realistic environment.

Finally, For the case of fair queuing networks deploying the packet-pair approach with MLDA for the bandwidth estimation part would improve the performance of MLDA in terms of stability and maintain its flexibility in heterogeneous multicast environments.

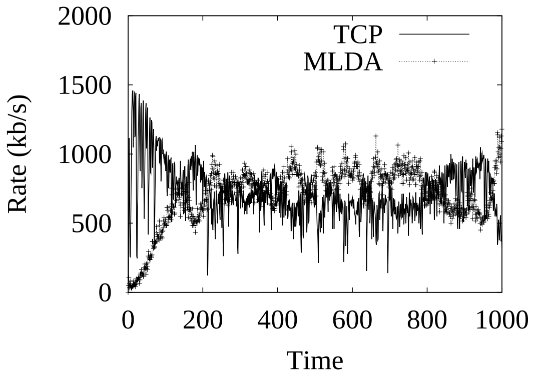
6.5.2 Comparison of MLDA and RLC

Vicisano et al. present in [191] a scheme called receiver-driven layered control (RLC) for realizing TCP-friendly congestion control. With RLC, the sender divides its data into layers and sends them on different multicast sessions. To test the availability of resources, the sender periodically generates short bursts of packets followed by an equally long relaxation period in which no packets are sent. For the duration of the bursts the consumed bandwidth by the flow is doubled. Hence, if the packets of a burst are not lost then this indicates the availability of resources. After receiving a packet burst, the receivers can join a higher layer if the burst was lossless otherwise they remain at their current subscription level. The receivers might leave a layer at any time if losses were measured.

In [191] a simulation topology is used similar to Fig. 6.14 but with the round trip delay (τ) set to 0.42 seconds and the bottleneck bandwidth (R) set to 1.5 Mb/s. The link is shared between 8 TCP connections and 8 adaptive flows and the packet size is set to 1024 bytes. Fig. 6.16(a) suggests that RLC shows a very conservative behavior under these conditions and the RLC flows receive only around 60% of the bandwidth share consumed by the TCP connections. The behavior of MLDA in this case resembles the behavior of TCP to a greater extent with the MLDA flows receiving around 90% of the bandwidth consumed by the TCP connections, see Fig. 6.16(b).



(a) RLC



(b) MLDA

Figure 6.16: Bandwidth distribution with RLC and MLDA

In a second test, we compared the behavior of RLC and MLDA in a multicast tree with shared links, see Fig. 6.17. This topology was similar to that described in Sec. 6.4.4 but with different delay and bandwidth parameters. Also in this case, the routers were shared between the adaptive flow and uncorrelated traffic.

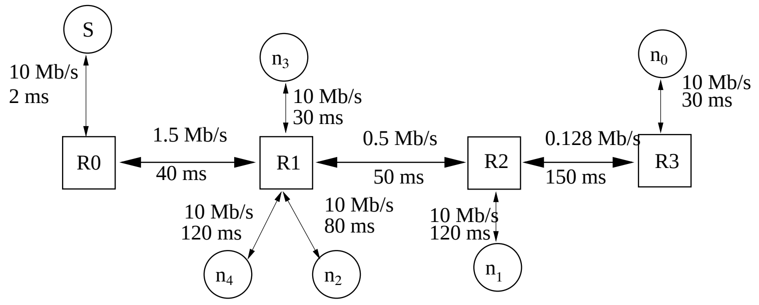
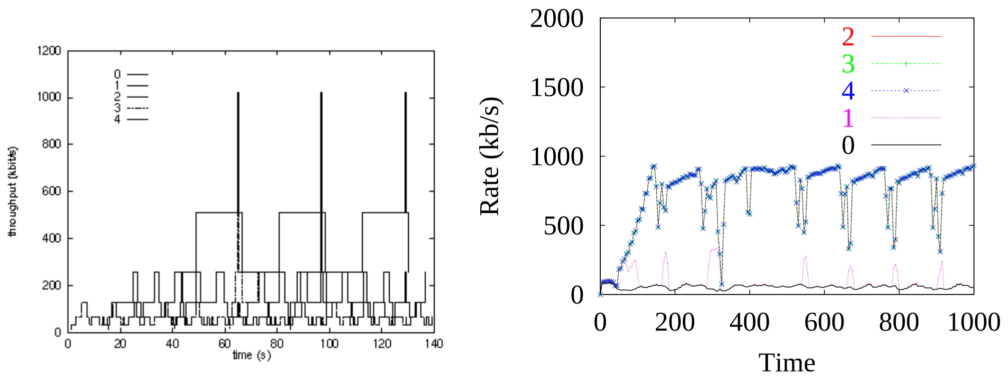


Figure 6.17: Testing topology of shared multicast trees

As Fig. 6.18 describes, receivers connected to the same bottleneck receive identical shares and stay synchronized throughout the simulation time with both RLC and MLDA. However, with RLC the maximum bandwidth share of the receivers (n_2, n_3, n_4) is restricted by the static nature of the transmitted layers. That is, while the background traffic only consumes half of the 1.5 Mb/s, the receivers (n_2, n_3, n_4) can only utilize up to 500 kb/s



(a) RLC (b) MLDA

Figure 6.18: Behavior of RLC and MLDA in a shared multicast tree

instead of 750 kb/s. With MLDA the sizes of the layers is shaped in accordance with the available resources and the receivers (n_2, n_3, n_4) receive on the average a bandwidth share of around 750 kb/s. Hence, while MLDA introduces a higher complexity due to the exchange of control messages between the sender and the receivers it is more capable of accommodating the needs of heterogeneous receivers than RLC.

6.5.3 Comparison of MLDA and IRFMC

Jiang et al. [86] present a scheme called inter-receiver fair multicast communication in which the sender transmits its data in a fixed base layer and a variable enhancement layer. Based on their measured losses, the receivers estimate their bandwidth share and report this to the sender. The transmission rate of the variable layer is then determined as to increase the satisfaction of most of the receivers.

Receivers determine their appropriate bandwidth share by measuring the losses of the incoming flow. In case no losses were observed the receivers increase their estimations of their bandwidth share by $(\frac{r_{\text{current}}}{l_{\text{tol}}})$ with r_{current} as the current transmission rate of the session and l_{tol} as the loss that can be tolerated by the receiver. In case of losses, the receivers reduce their bandwidth estimation by a reduction factor intended to be similar to TCP's rate reduction in face of losses.

To compare the behavior of IRFMC with MLDA we chose a test topology depicted in Fig. 6.19 as was described in [86].

The TCP connection was modeled as a Tahoe-TCP [179] connection and started at 80 seconds and stopped transmitting data at 250 seconds. The routers in the topology used the drop tail buffer management and had a buffer of 20 packets. The multicast flow started transmitting data with a rate of 200 kb/s.

As Fig. 6.20(b) depicts, with MLDA the sender starts increasing the transmission rate of its base layer until it reaches the capacity of router R2. To further utilize the

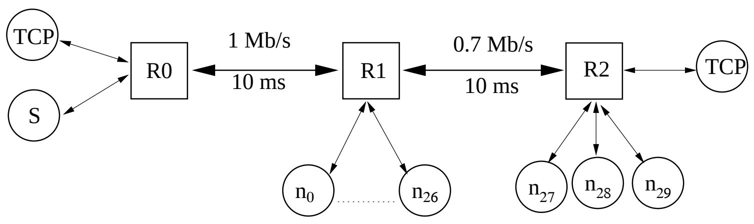


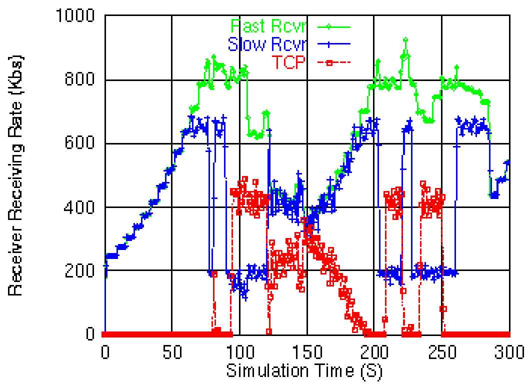
Figure 6.19: Testing topology for MLDA and IRFMC

resources still available at router R1, the sender starts increasing the transmission rate of the enhancement layers. When the TCP connection starts transmitting data, the sender reduces the transmission rate of the base layer to allow for fair bandwidth distribution between the TCP connection and the slow receivers with each of the TCP and MLDA flows receiving a bandwidth share of 350 kb/s at router R2. Receivers connected to the faster router get a bandwidth share of around 600 kb/s during the active time of the TCP connection.

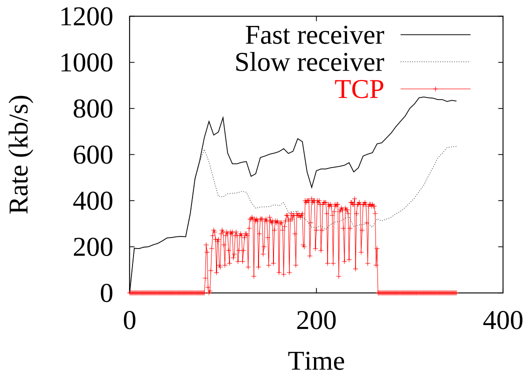
With IRFMC, the TCP connection achieves an average bandwidth share of 220 kb/s [86]. However, from Fig. 6.20(a) we can see that the TCP connection receives a bandwidth share of around 400 kb/s in the first half of its life time. In the second half its share is reduced considerably and it only receives a share of the bandwidth when the slower receivers temporarily leave the higher layer.

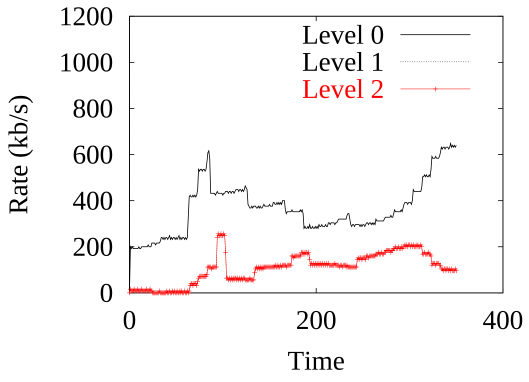
6.6 Summary

To accommodate the case of multicast communication in heterogeneous environments a general congestion control scheme called MLDA was presented in this chapter. With MLDA the receivers determine the appropriate TCP-friendly bandwidth share on the path connecting them to the sender and inform the sender abut this value. The sender uses this information to periodically adjust its transmission rate in accordance with the network congestion state. To take the requirements and capabilities of heterogeneous receivers into account MLDA senders use layered data transmission with the size and number of layers determined dynamically in accordance with the receiver reports. In addition to the general architecture for regulating the behavior of the receivers and senders, this chapter described an approach for scalable round trip delay measurements as well as a feedback scheme that avoids the feedback explosion problem on the one hand but provides the sender with information about the different capacities of the receivers on the other. To use RTP for exchanging control information between the sender and the receivers various enhancements to RTP had to be introduced in terms of the transported control information as well as the method for setting the transmission frequency and periods of the control messages. This was necessary due to the poor results achieved with the current specification of RTP in terms of timely and accurate information exchange between the sender and the receivers for the case of large multicast groups as described in Sec. 6.1. Results obtained by simulating MLDA over different test topologies as well



(a) IRFMC





(b) MLDA: Receivers (c) MLDA: Levels

Figure 6.20: Bandwidth distribution with IRFMC and MLDA

as by comparing it to a number of other schemes suggest the efficiency of MLDA in terms of TCP-friendly bandwidth distribution and accommodating the needs of large and heterogeneous multicast groups.

Chapter 7

Network-Assisted Congestion Control

In the previous chapters, we presented adaptation schemes in which the end systems measure and estimate the performance of the network and determine the appropriate bandwidth share to use based on these estimations. However, such mechanisms can only deliver estimations of the network load state and not the actual available resources. As information about the actual availability of network resource is only available at the network nodes, we present in this part of the work, an approach in which the network nodes inform the end systems about the available resources. With this approach called adaptive load service (ALS), the sender transmits control messages including its desired transmission rate to use. The intermediate routers adjust this value in accordance with their available resources and forward the control messages to the next network node until they reach the receiver. The receiver in its turn transmits the updated information back to the sender who needs to adjust its transmission behavior in accordance with the received information. As network routers have more detailed information about the resource availability, with ALS we achieve a smoother adaptation behavior and realize max-min fair bandwidth distribution.

Like MLDA, ALS can use the RTP/RTCP [158] protocol for carrying the control information. To accommodate the needs of multicast communication in heterogeneous environments, ALS supports layered transmission with the sizes and numbers of layers determined using the information collected from the network nodes and receivers involved in the communication. In this case, ALS provides the receivers with exact information about the number and sizes of layers to join in order to receive the QoS level that corresponds with the network capacities.

In Sec. 7.1, we discuss some of the problems related to end-to-end adaptation. ALS is then described in Sec. 7.2. Sec. 7.3 discusses some of the issues related to using ALS in the Internet like processing overhead and control protocol. In Sec. 7.4, we investigate the performance of ALS and its behavior under different network topologies.

7.1 Motivation

With the loss-delay based adjustment scheme (LDA), we presented in chapter 4 an approach for adapting the transmission rate of a sender based on the feedback information from the receiver about the congestion situation in the network. In chapter 6, we presented a general framework for TCP-friendly congestion control for multicast communication in heterogeneous environments. With this framework, called MLDA, the sender transmits its data in layers. The size and number of layers is determined by the receivers' estimation of the congestion situation on the paths connecting them to the sender and the appropriate transmission rates to use on those paths. The congestion avoidance is realized with MLDA by the join and leave actions of the receivers who change the number of the layers they want to listen to based on their estimation of the congestion situation in the network. As the number of layers transmitted towards a receiver determines the amount of data carried in the network, joining a new layer increases the load in the network and leaving a layer usually results in a load reduction.

With end-to-end adaptation schemes, the end systems, i.e., the sender and the receivers, cooperate in collecting information about the network congestion state, estimating the available resources and adapting the amount of data entering the network. In this end-to-end scenario the network plays only a passive role of transporting the data between the sender and the receivers.

End-to-end approaches have the clear advantage of simplicity. As they require no changes to the network, they can be introduced instantaneously and hence be gradually deployed without having to wait for support from the network providers. However, with end-to-end measurements end systems can only get an estimation of the characteristics of the network, i.e., the bottleneck bandwidth, delay and losses between the end systems and not the actual available resources for a flow. Based on the measured characteristics the end systems can probe the network in order to estimate the available resources for the data flows. That is, if the measured characteristics indicated an underload state, the flow's bandwidth share can be increased. However, as the end systems have no knowledge about the available resources, they might increase the flow's bandwidth share beyond the actual available resources and hence cause congestion. On the other hand, network overload can only be indicated by data losses or increased delays. That is, the end systems react to the overload situation when it is already too late. Also with no knowledge about the actual available network resources the end systems might reduce their consumed bandwidth to a level that is still too high for the network and hence still causes losses or to a level that is too small and thus underutilizes the network.

In addition to the probing problem, adaptation schemes based on end-to-end measurements and bandwidth share estimations show severe fairness problems in terms of the max-min fairness definition as was presented in Sec. 3.1.1.4. For flows having the same requirements and characteristics but traversing a different number of links TCP [51, 171] and older ABR schemes based on bit marking [169] show severe unfairness towards flows traversing a higher number of links. This is due to the beat-down problem [169] that arises as a result of the increase in the loss probability with each additional traversed link. Hence, the flows traversing a larger number of links face higher loss values than those going over shorter distances. Hence, those long distance flows decrease their sending rates more often and with higher values than the short distance ones. Until now, no viable solution has been proposed for solving this problem in TCP. In the case of ABR, several solutions requiring the network switches to calculate the fair share and sending this value to the sources were introduced [81], see App. A.

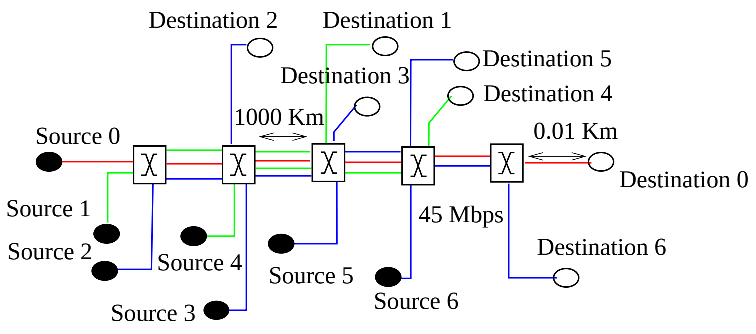


Figure 7.1: Chain topology for testing max-min fairness

To test the effects of the beat down problem on LDA we use a chain topology with five routers and 7 sources each generating and sending an LDA controlled flow towards a receiver, see Fig. 7.1. The distance between two neighboring routers is 1000 km and the distance from a source to a switch is 0.4 km. All the links have a bandwidth of 1 Mb/s. The sending sources use an initial additive increase rate ($\dot{R}_{\mathrm{ai}}$) of 5 kb/s, Depending on the number of routers they have to pass, three kinds of traffic can be distinguished:

- Long distance flows: The flow starting from source 0 traverses all four links. This flow will be denoted as C_0.
- Medium distance flows: The flows starting from sources 1 and 4 traverse two links. These flows will be denoted as C_1 and C_4.
- Short distance flows: The flows starting from sources 2, 3, 5 and 6 traverse only one link. These flows will be denoted as C_2, C_3, C_5 and C_6.

Connections	C_0	C_1	C_2	C_3	C_4	C_5	C_6
fair share (kb/s)	250	250	500	250	250	500	750
achieved share (kb/s)	85	158	617	494	160	641	872

Table 7.1: Expected and achieved bandwidth shares for the single flows using LDA

Tab. 7.1 presents the bandwidth shares we would expect using the max-min fairness definition for bandwidth distribution, see Sec. 3.1.1.4, and the actual average transmission rates of the different senders. We observe that the flows traversing more than one router, receive a much smaller bandwidth share than flows traversing only one. Using Eqn. 3.3 we get a fairness index of 0.80 which is a rather bad fairness indication. This is especially evident for the long distance flow (C_0) which receivers less than 30% of its theoretical fair share. The medium distance flows (C_1, C_2) receiver around 50% of their theoretical share.

7.2 Specification of the Adaptive Load Service (ALS)

When designing an adaptation mechanism for QoS control we usually need to consider the following goals:

- achieve an overall high resource utilization,
- reduce losses,
- maintain a stable quality-of-service,
- distribute the available resources in a fair way,
- scale to large and heterogeneous multicast groups.

Notice that while the issue of fair resource distribution is of utmost importance it is a subjective measurement. Similar to our approach in the previous chapters, we will consider a system to be fair if similar connections receive similar shares. Additionally, for the case that different connections have the same characteristics and requirements but differ in the number of traversed network hops, our goal will be to achieve the max-min fairness criterion, see Sec.3.1.1.4.

The adaptive load service (ALS) was designed in a similar fashion to the ABR service in ATM. That is, the senders transmit control packets containing information about their desired amount of resources and the network nodes calculate the bandwidth shares the flows should be using and include them in the control messages. The receivers, finally, send the adjusted information back to the senders which need to adjust their transmission behavior based on the received control information. To accommodate heterogeneous receivers, the sender transmits its data in layers with the layer sizes and numbers based on the feedback information from the receivers similar to MLDA. For distributing control information, the real time transport protocol (RTP) [158] with enhancements similar to those presented in Sec. 6.3 are used.

7.2.1 Sender Behavior

Similar to MLDA, an ALS sender sends every $(T_{\text{control}} + p)$ seconds a sender report with T_{control} as a constant and p as a random variable that is used to avoid synchronization among senders starting at the same time.

The control packets include a field for indicating the sender's desired transmission rate field (R_d) and a network utilization field (U) that is initialized to 0. In case the desired rate is not known, R_d can be set to the maximum possible rate, i.e., the rate of the output link of the sending end system.

Routers traversed by the control messages update the value of the desired rate in accordance with their own resources and forward them to the next hop. The receivers determine the fair share the sender should be using and inform the sender about this value using the receiver reports.

In between sending control messages the sender collects the receiver reports and based on the rate information reported by the receivers determines the number and sizes of the layers to use in a similar fashion to MLDA, see Sec. 6.2. The information about the number (k), size ($[R_{\mathrm{L}_1}, \cdots, R_{\mathrm{L}_k}]$) and addresses of the used layers are also included in the control messages.

7.2.2 Router Behavior

The senders transmit control messages containing fields indicating the desired rate (R_d), i. e., the rate the sender wishes to use, and a network utilization field (U).

In addition to the usual functionalities provided by a router such as buffer management and routing, ALS capable routers need to be enhanced with functionalities for packet and flow accounting and service handling, see Fig. 7.2.

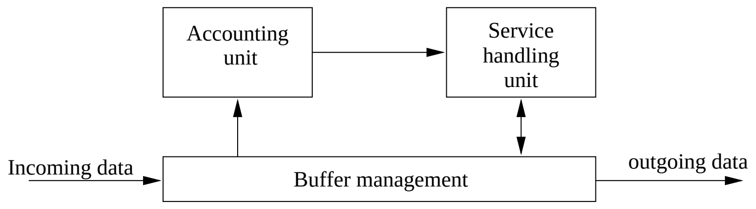


Figure 7.2: Router components for ALS

Service handling unit: This unit investigates the content of the arriving control packets and compares the indicated desired rate value (R_d) of the flow with a locally calculated fair bandwidth share this flow should be using (R_f). For the case of $R_d \geq R_f$, R_d is set to R_f and the control packet is forwarded with the new value to the next hop. Additionally, to provide the end systems with a more complete view of the network congestion state the router includes its utilization level in the U field for the case of $R_d \geq R_f$. The utilization level is determined in the accounting unit.

A large part of the work done in the context of the ABR service was dedicated to the issue of estimating the appropriate fair bandwidth share to advertise at the switches. This is basically done using the content of the control cells sent by the senders, which indicates the desired rate the sender would like to use and the current rate the sender is using. Based on this information, the switches can estimate the number of connections congested at another network node and the available bandwidth share to distribute among the competing ABR connections. Even with efficient algorithms such as [89, 82, 10] various state information has to be saved and several multiplication operations and data lookups are needed. Depending on the complexity of the scheme and the amount of saved state information about the connections, several estimation cycles are needed to calculate an appropriate and stable fair bandwidth share. For the case of ABR, each estimation cycle consists of the round trip delay and the time between sending two control cells which is usually set to the time needed for sending 32 ATM cells. However, such a high frequency of control messages would result in the case of IP in significant overhead in terms of link bandwidth and processing resources at the routers. Hence, it is expected that control messages would be sent every a few seconds and in case RTP was used for the distribution of control information T_{control} would be in the range of five seconds and more. So with around 10 estimation cycles required for determining the fair share, which is an appropriate value for a moderate complexity algorithm [164], ALS would need around one minute until determining the fair share. Additionally, as the available bandwidth and number of flows will most probably change during the time needed for determining the fair share, the estimation time will even increase further. Hence, using a distribution algorithm similar to those proposed for ABR would not be of much benefit and would only lead to more complex routers. Therefore, ALS routers determine the fair rate share (R_f) as follows:

$$R_f = \frac{R \times U_l}{N_{\text{conn}}} \tag{7.1}$$

with R indicating the bandwidth dedicated to the adaptive flows and N_{conn} as the current number of active flows, whereas a flow is recognized through the IP-header with the source and destination addresses. U_l represents the utilization level we would like to achieve. Notice that setting U_l too high would lead to high utilization but also to high losses for the case of sudden changes in the number of flows or available bandwidth. In this work, we use a utilization level (U_l) of 80% which is equivalent to the value used in similar switch mechanisms for ABR [10].

Accounting unit: For determining the appropriate bandwidth share the end systems should be using, ALS routers require knowledge about the number of active flows and the end systems need information about the utilization level of the routers.

The utilization level (U) is measured as the number of packets (N_{packets}) passing a router over an observation time period ($T_{\text{observation}}$) compared to the capacity of the router (R).

$$U = \frac{N_{\text{packets}}}{R \times T_{\text{observation}}} \tag{7.2}$$

To smooth out sudden changes in the utilization level, the router includes in the control packets an averaged value of the utilization (U_{average}) that is determined as follows:

$$U_{\text{average}} = (1 - \alpha) \times U_{\text{average}} + \alpha \times U \tag{7.3}$$

in this work, we set α to 0.5 which gives a good balance between the effects of the last value of U_{average} and the new value of U on the final value of U_{average}.

Determining the number of flows (N_{conn}) traversing a router is, however, more difficult than determining its utilization. A large fraction of the Internet flows today consists of short transfers of a few data packets as is the case of WWW and DNS [117] traffic. Depending on the rate at which the end systems receive information about the bandwidth share they should be using, counting the short lived flows differs substantially. For the case that the end systems use frequent control messages in the order of one every few round trip delays, i.e., the average life time of a short lived flow, long- and short-lived flows can be treated as equals and the available router capacity needs to be equally divided among all flows. However, such an approach requires a large number of control messages and subsequently a high processing overhead for reading and processing those control messages at the routers. Additionally, this would lead to a rather oscillatory behavior at the end systems. For the case that the end systems receive infrequent updates on the available network resources, dedicating equal bandwidth shares for long- and short-lived flows would reduce the overall utilization. For example, with ALS control messages are transmitted in periods of a few seconds (T_{control}) and, hence, the end systems can only adapt to changes in the bandwidth availability in periods of T_{control}. In such a case, the long-lived flows would be restricted to their share for a period of T_{control}, whereas the short lived flows would not be able to fully utilize their own share for this entire time. On the other hand, fully neglecting the short-lived flows would mean that all the bandwidth is dedicated to the long-lived traffic and would be unfair towards the short ones.

In the work presented here, N_{conn} is set to the number of flows that were active during two consecutive observation periods ($T_{\text{observation}}$). That is, during each observation period the ALS router lists all flows from which data was received. At the end of the observation period this list is compared to the list collected in the previous observation period and only the flows that were listed during the two periods are considered as active flows. On the one hand, this approach counts all long lived-flows and some of the short-lived ones that start at the end of one observation period and end during the second period. On the other hand, it ignores flows that start and end during the same period. Hence, the bandwidth is not fully dedicated to the long-lived flows without distributing it equally among all flows. Notice that this approach is just an example of a flow counting algorithm. Depending on the actual numbers of the short- and long-lived flows, the relative priority of one compared to the other or user preferences any other counting mechanism could used.

Finally, to smooth out sudden changes in the observed number of flows, the router uses an averaged value of the flow count (N_{conn}) that is determined as follows:

$$N_{\text{conn}} = (1 - \alpha) \times N_{\text{conn}} + \alpha \times N \tag{7.4}$$

with N as the number of flows determined after an observation period. In this work, we set α to 0.5 which gives a good balance between the effects of the last value of N_{conn} and the new value of N on the final value of N_{conn}.

7.2.3 Receiver Behavior

After traversing all the intermediate routers the control messages contain in the R_d field the smallest possible fair share (R_f) calculated at all traversed network nodes and the utilization of the router where this fair share was determined.

Notice that R_f does not necessarily need to be equivalent to the bandwidth share determined using the max-min fairness criteria. As an example, consider the topology depicted in Fig. 7.3. The first router has a capacity of 2 Mb/s and is shared between 2 connections, hence, the router would calculate R_f to be 1 Mb/s. However, the second router traversed by the connection from sender 1 to receiver 1 has only a capacity of 1 Mb/s and is also shared between two connections, and thus the fair share value that will finally be advertised to sender 1 is only 0.5 Mb/s. Now, if sender 2 restricted its transmission rate to the fair share value included in the control messages, i.e., 1 Mb/s, 0.5 Mb/s of the bandwidth available at router 1 will be wasted.

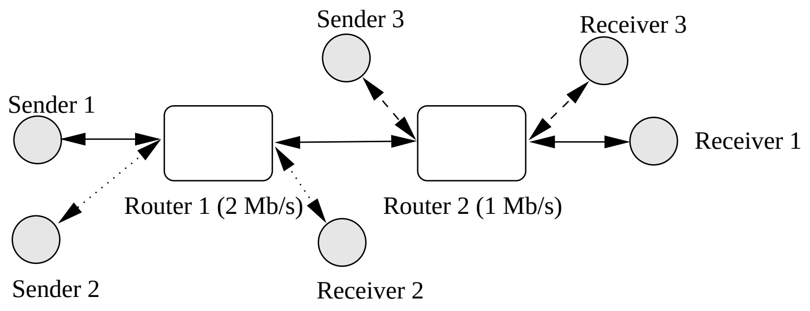


Figure 7.3: An example for max-min distribution

To compensate this deficiency in the share calculation of the router scheme but still preserve its simplicity, the receiver estimates in a similar manner to LDA an actual bandwidth share to use and reports this value to the sender in its reports.

The receiver determines the actual bandwidth share (R_{actual}) based on the network load situation observed in the period prior to receiving a sender control message. Now consider that the mth. control message (M_{m}) arrives at the receiver. Based on the loss values measured in between receiving M_{m} and $M_{\text{m-1}}$, the utilization level ($U_{\text{m-1}}$) as reported in $M_{\text{m-1}}$ and layer sizes ($[R_{\text{L}_1}, \cdots, R_{\text{L}_k}]$) reported in $M_{\text{m-1}}$ the receiver determines R_{actual_m} as follows:

No loss situation: Based on the assumption that the reported R_d value might be less than the fair bandwidth share that would have been calculated with an algorithm using the max-min fairness criterion, the receiver can try to increase its bandwidth share. Similar to LDA the rate is increased additively with an additive increase rate (R_{ai}) with R_{ai} determined in relation to the utilization level reported in $M_{\text{m-1}}$

$$R_{\text{ai}} = R_{\text{ai}} + (R_{\text{ai}} \times (1 - U_{\text{m-1}})) \tag{7.5}$$

Now as its is possible that network routers have already considered this underload situation in their latest share calculations, the receiver takes the maximum of the network determined share (R_{d_m}) and its own estimated share. The actual rate (R_{actual_m}) is then set to

$$R_{\text{actual}_m} = \max(R_{\text{actual}_{m-1}} + R_{\text{ai}}, R_{\text{d}_m}) \tag{7.6}$$

with $R_{\text{actual}_{m-1}}$ indicating here the actual bandwidth share the receiver has determined based on the measured losses and the information in $M_{\text{m-1}}$.

Loss situation: If losses were measured in between the reception of two control messages, then R_{actual_m} is set to R_{d_m} and R_{ai} is reset to an initial value ($\dot{R}_{\text{ai}}$).

Using R_{actual_m} the receiver can now determine the number of layers it can join similar to MLDA, see Sec. 6.2.2. That is, depending on the value of R_{actual_m} and the sizes of the transmitted layers the receiver can listen up to x layers while

$$R_{\text{actual}_m} \geq \sum_{k=0}^{x} R_{\text{L}_k} \tag{7.7}$$

with R_{L_k} as the rate of layer k as indicated in message M_{m}. Finally, the receiver schedules the transmission of a report carrying the value of R_{actual_m} in the R_d field after a time period of T_{wait}.

To reduce the number of receiver reports received by the sender for the case of large multicast groups, ALS is, similar to MLDA, based on the partial suppression approach, see Sec. 6.2.3. With this approach, the receiver partitions the possible range of bandwidth into intervals and suppresses the transmission of its own reports if prior to the expiration of T_{wait} a receiver report from some other receiver was seen carrying a rate value that is similar to R_{actual_m} this receiver has just determined.

7.3 Implementation Issues

While Sec. 7.2 describes the general specifications of ALS, this section discusses some of the practical aspects of implementing and using ALS. In this context, we discuss the issues of parameter settings of ALS and the possibility of using RTP for distributing control information.

7.3.1 Distribution of Control Information

Again similar to MLDA, a control protocol is required to allow for the cooperation and exchange of information between the sender and the receivers. RTP already offers a wide range of functionalities for identifying data flows and transporting control information about losses and used bandwidth. To use RTP as the signaling protocol for ALS similar enhancements to those introduced for MLDA are required:

Control information: The ALS sender needs to inform the receivers about the sizes, addresses and number of transmitted layers. This information can be included in the sender RTCP reports as an application specific part (APP). Similarly, the receivers can include in their RTCP reports their estimated bandwidth shares as further applications parts.

Control periods: In contrast to RTP, ALS senders generate control messages in fixed periods. After receiving a sender report the receivers schedule a feedback report and use the partial suppression approach for reducing the total number of reports. As already described in Sec. 6.3 this part of the RTCP specification needs to be adjusted to allow for a more scalable and timely flow of feedback information from the receivers to the sender.

7.3.2 Processing Overload of RTCP Messages

To alert transit routers to more closely examine the contents of the control messages, the senders' RTCP messages could be carried in IP packets with the IP router alert option [92]. This is the same approach as was proposed for RSVP or YESSIR [131].

After receiving a control message, ALS routers set the appropriate R_d and U values in the RTCP messages and forward the packet to the next hop. To avoid the need to recalculate the UDP checksum the sender should set it to 0.

To test the additional load needed for updating the fair share values in the RTCP messages in the routers we implemented a simple ALS process that reads RTCP packets from an open socket, parses the RTCP packets, checks if the included fair share value is lower than some local value, writes the local fair share into the RTCP packet and forwards the packet to another station.

At the sender station we used a tool that sends a burst of a 1000000 RTCP packets each with the length of 156 Bytes to the station with the ALS process over an ATM link. The station running the ALS process is a SUN Ultra 1 with a 137 MH processor. Measured with the UNIX command *top* the ALS process consumed around 45% of the station's processing power to read, process and forward RTCP packets with a rate of around 1100 packets/second. This would suffice for an OC-3 link carrying 5000 audio flows of 32 kb/s and sending an RTCP packet each 5 seconds.

7.3.3 Parameter Settings

Similar to MLDA, with ALS the senders generate control messages every $(T_{\text{control}} + p)$ and the receivers need to wait for a period of T_{wait} before sending their reports. As discussed in Sec. 6.2.7 setting T_{wait} to 1.5 seconds results in a good degree of suppression. p can be arbitrarily set to 0.5 seconds. Now considering a maximum round trip delay of one second, to account for the time required for the receiver reports to arrive at the sender, T_{control} can have any value higher than three seconds. To maintain the similarity to RTP and reduce the processing and bandwidth overhead of more frequent control messages we set T_{control} in this study to 5 seconds. Notice that as with ALS there is no need for measurement periods as was the case with MLDA the interval from which T_{wait} can be determined from can be larger than with MLDA and would thus provide for better suppression and fewer receiver reports.

7.4 Performance of the Adaptive Load Service

In this section, we investigate the performance of the ALS scheme under different simulated topologies. In particular, we investigate the performance of ALS in terms of achieved utilization, avoiding losses and its fairness towards other ALS connections as well as competing TCP traffic. Additionally, we investigate the behavior of ALS in heterogeneous environments and for the case of multiple congested hops.

7.4.1 Competing TCP and ALS Traffic

For any adaptation scheme to be effective in the Internet environment it needs not only to ensure high utilization, avoid losses but also be fair towards competing TCP traffic.

To test the TCP-friendliness of ALS we used the same simulation topology used for evaluating LDA, see Fig. 2.5, with a bottleneck link of 10 Mb/s shared between m ALS flows, n FTP connections and with k WWW servers initiating TCP connections in an on-off approach as was described in Sec. 2.3.1. The used router was a RED router, see Sec. 2.1.1.1, which ensures equal loss distribution and reduces synchronization effects. The minimum buffer threshold of the RED router was set to 30% of the available buffer and the maximum threshold to 80%. The queuing weight (w_q) was set to 0.002 and the maximum drop probability (P_a) to 0.02. Finally, the packet size was set to 1000 bytes and the maximum queuing delay (τ_{q}) was set arbitrarily to 0.1 seconds.

The ALS router used an observation period ($T_{\mathrm{observation}}$) of two seconds for determining the utilization level and the number of active flows. The value of $T_{\mathrm{observation}}$ was rather arbitrarily chosen as we had run a number of simulations with $T_{\mathrm{observation}}$ having different values in the range of one to five seconds without observing any significant difference in the performance of ALS. The maximum utilization level (U_l) was set to 0.8. The initial additive increase rate ($\dot{R}_{\mathrm{ai}}$) was set to 5 kb/s.

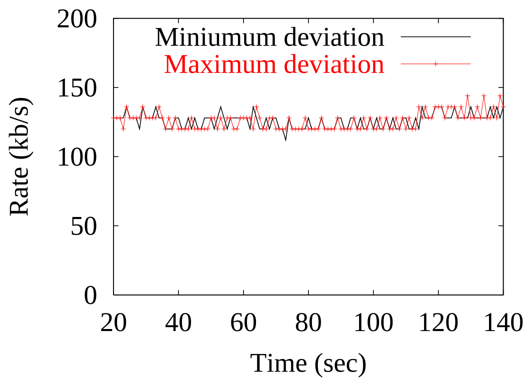
7.4.1.1 Basic Performance Tests

Similar to the work presented in Sec. 4.4.2.3 we tested in this part the performance of ALS in the presence of competing FTP and WWW flows for the case of different round trip delays and number of competing flows. As background traffic we used 27 WWW servers generating short bursts of TCP traffic in an on-off process as was described in Sec. 2.3.1.

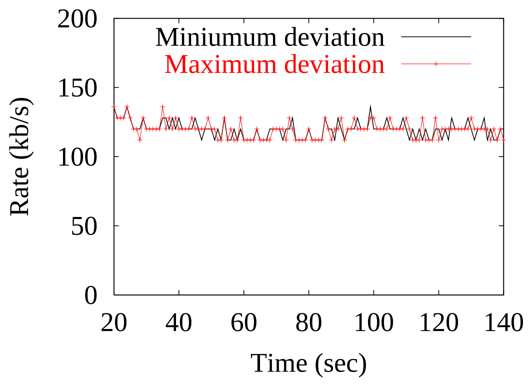
N	27					54				
$\tau(sec)$	r (kb/s)	σ (kb/s)	$\overline{\sigma}$ (kb/s)	F	u	r (kb/s)	σ (kb/s)	$\overline{\sigma}$ (kb/s)	F	u
0.1	127	6	0.28	0.63	0.87	69.9	5.1	0.08	0.64	0.90
0.2	119	6.4	0.2	0.58	0.87	67	4.4	0.1	0.62	0.88
0.4	114	5.3	0.2	0.62	0.8	66	4.4	0.1	0.61	0.91
1.0	110	5.2	0.2	0.59	0.83	63	4.7	0.1	0.66	0.87

Table 7.2: Achieved average rate (r), utilization (u) and fairness (F) of ALS with N TCP, N ALS competing flows and 27 WWW servers, R set to 10 Mb/s and τ_q set to 0.1

Tab. 7.2 depicts the results for the average rate of the ALS flows (r), the standard deviation of each flow from its own average (σ), the deviation of the average flow rates from the overall average ($\overline{\sigma}$) as well as the utilization level (u). The results of the simulations present a friendliness index (F) of less than one suggesting that ALS is rather too conservative in its adaptation behavior. However, the adaptation behavior of ALS is primarily driven by the bandwidth share estimates of the network nodes. With the flow counting mechanism described in Sec. 7.2.2 the router counted on the average 65 flows each observation interval for the case of ($N = 27$). Hence, the bandwidth share of the ALS flows should be ($\frac{27}{65}$) of the available bandwidth. While the ALS flows restricted their bandwidth shares to the level indicated by the router, the long-lived TCP connections managed to increase their share and accumulate the share that was designated for the short WWW flows. For ALS flows to receive a higher share, another counting mechanism needs to be deployed that ignores short-lived flows and a higher utilization level (U_l) should be allowed.



(a) $N = 27, \tau = 0.2, \tau_q = 0.1$



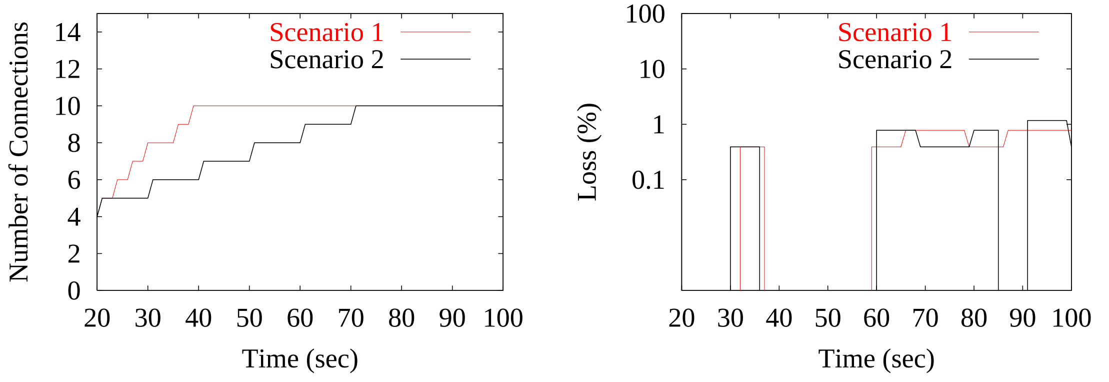
(b) $N = 27, \tau = 0.4, \tau_q = 0.1$

Figure 7.4: Temporal behavior of ALS

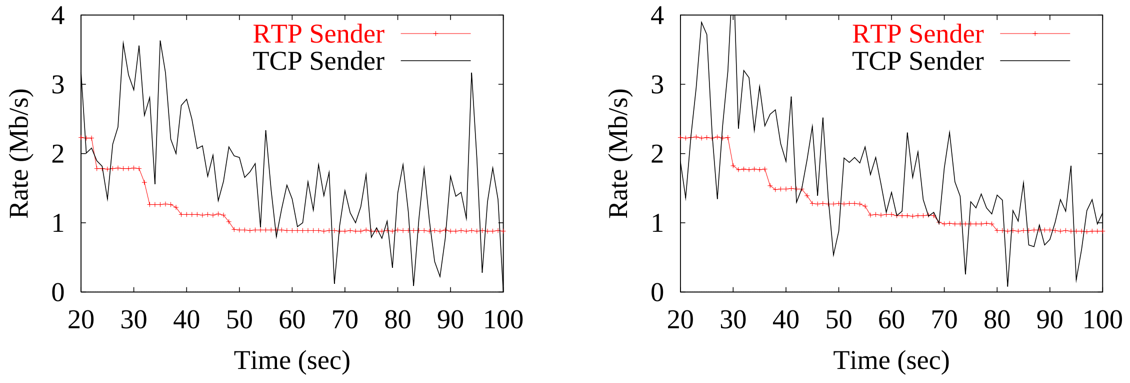
With ALS the deviation in the average bandwidth share among the different flows is rather negligible indicating a high inter-flow fairness with all the flows receiving similar bandwidth shares. The deviation value of the measured rate of the single flows from their own average is in the range of 5%. Fig. 7.4 depicts the temporal behavior of two ALS flows with the largest and smallest deviation values (σ) for different round trip delay (τ) values. Compared to Fig. 4.21 and Fig. 5.3 which display the results for flows using LDA and CTFAF under the same network parameters we can observe that with ALS the flows have a stable bandwidth share and each flow receives its bandwidth share as determined by the network during the entire simulation time without observable oscillations. Finally, notice that the bandwidth shares of ALS flows do not change significantly with increasing round trip delays and that the fairness factor (F) is not influenced by the delays in the network.

7.4.1.2 Performance of ALS with Changing Numbers of Flows

In this case, we investigated the effects of changing the number of ALS flows on the bandwidth distribution. We considered two scenarios: in the first scenario, the number of ALS flows increased with the time interval between the starting points of two flows set to less than the interval between two subsequent control messages of T_{control} which was set here to five seconds. In the second scenario, the flows were started at intervals larger than T_{control}. In both scenarios, the link bandwidth is set to 10 Mb/s and the round trip propagation delay to 0.01 seconds. The number of TCP connections was set to 3. Fig. 7.5 shows that even though the addition of connections in this topology causes at



(a) Number of connections (b) Loss measurement



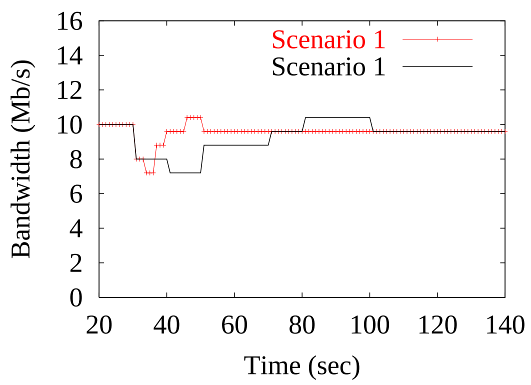
(c) Scenario 1 (d) Scenario 2

Figure 7.5: Performance of ALS with varying number of competing connections

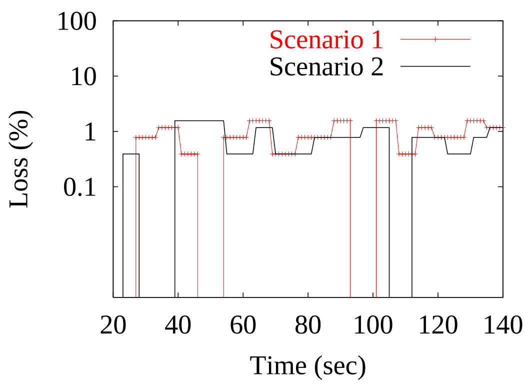
least a 10% increase in the number of connections at the routers the transmission rate of a sender starting at time 0 changes in accordance with the new network conditions for both scenarios without facing high losses, see Fig. 7.5(b). This test also shows the importance of choosing an appropriate value for the utilization factor at the routers. Choosing a high value would not account for sudden changes in the network conditions, i.e., the number of connections. As the ALS connections need a few seconds until receiving a new value of the updated fair bandwidth share to use, any increase in the number of connections would result in losses until the senders adjust their transmission rate.

Fig. 7.5(c) and Fig. 7.5(d) also show that the TCP connections receive a fair bandwidth share during the entire simulation time. Actually, they even receive a share that is around 10% higher than the share used by the ALS connections. This results from the shorter adaptation interval of TCP which allows the TCP connections to increase their transmission rate each round trip time and thus faster utilize any available bandwidth.

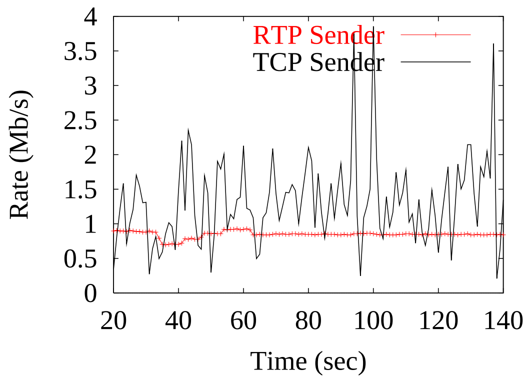
7.4.1.3 Performance of ALS with changing bottleneck bandwidth



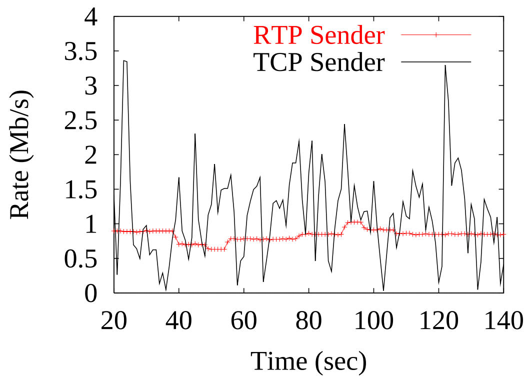
(a) Bottleneck bandwidth



(b) Loss measurement



(c) Scenario 1



(d) Scenario 2

Figure 7.6: Performance of ALS with varying bottleneck bandwidth

In this part, the bottleneck bandwidth was varied and the number of connections was held constant. We used three TCP connections and seven ALS flows. Again the variation was done in intervals smaller and larger than T_{control}. Just as in Sec. 7.4.1.2, Fig. 7.6 shows that the transmission rate of an ALS sender is adjusted in accordance with the available bandwidth and the losses are below 2%.

7.4.1.4 Interaction of ALS and WWW Traffic

To test the friendliness of ALS traffic towards short-lived WWW connections we can not simply compare the bandwidth shares of competing WWW and ALS flows. As WWW flows carry only a small number of packets before terminating they can not assume the same bandwidth share as a long-lived ALS flow. Therefore we quantify the friendliness of ALS traffic towards such short-lived flows by comparing the bandwidth share consumed by WWW flows generated by a number of servers when competing with long-lived TCP connections on the one hand and with ALS flows on the other.

In the simulation topology of Fig. 2.5, we simulated the situation of 27 long-lived flows competing with WWW connections generated by 54 servers that generated short TCP connections in an on-off process as described in Sec. 2.3.1.

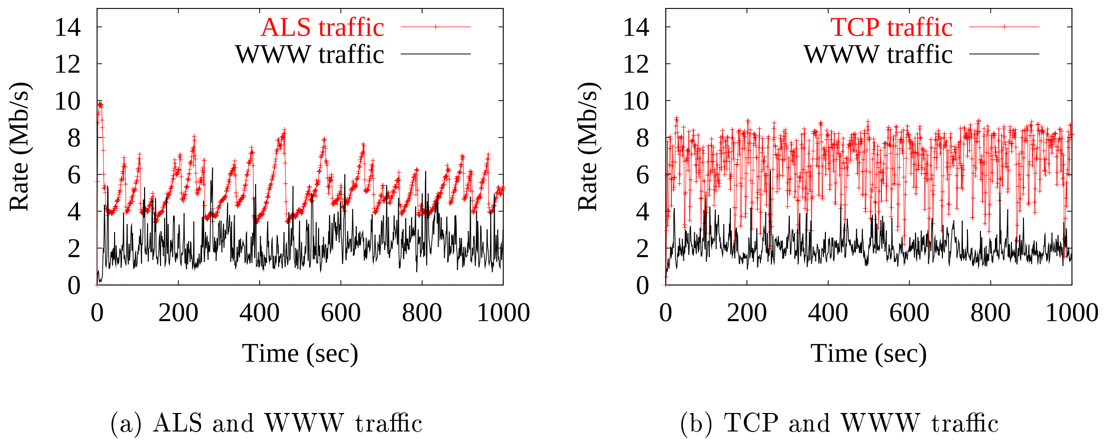


Figure 7.7: Bandwidth sharing for the case of WWW traffic competing with FTP and ALS flows

The bandwidth sharing results depicted in Fig. 7.7 suggest that the WWW traffic receives a similar share for both cases of competing with ALS or long-lived FTP traffic. For the case of competing with FTP traffic, the WWW traffic consumes a bandwidth share of 2 Mb/s. When competing with ALS traffic this share is increased to 2.2 Mb/s suggesting that ALS is even slightly more fair than long-lived TCP connections towards WWW traffic.

7.4.2 Performance of ALS in Terms of Max-Min Fairness

Already, there has been different proposals for adaptation schemes that achieve high utilization, low losses and are fair towards competing TCP traffic [173, 19]. However, just as it is the case for TCP [51, 171], all those algorithms show severe fairness problems when mixing long and short distance traffic and thus do not fulfill the requirements of the max-min fairness criterion [166]. For testing the performance of ALS in terms of the max-min fairness criterion we used the chain topology described in Sec. 7.1

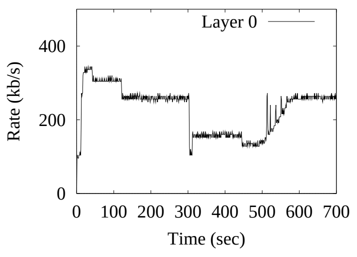
Connections	C_0	C_1	C_2	C_3	C_4	C_5	C_6
fair share (kb/s)	250	250	500	250	250	500	750
achieved share (kb/s)	248	249	497	245	246	499	742

Table 7.3: Expected and achieved bandwidth share for the single ALS flows in kb/s

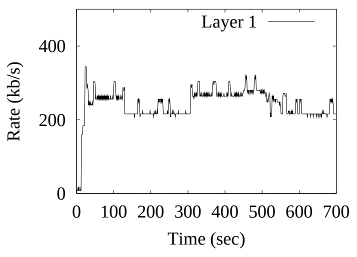
The results of the transmission rates used by the different connections shown Tab. 7.3 suggest that even though we used a simple distribution algorithm in the routers, the connections manage to achieve their fair bandwidth share based on the adaptation mechanisms of the end systems. The time required to achieve the fair bandwidth share is less than 100 seconds which corresponds roughly to 12 estimation cycles. While it is larger than what we would expect form an efficient ABR style algorithm [89] the gained router simplicity surely justifies this trade off. The fairness index in this case is 0.999 which indicates a very high degree of fairness.

7.4.3 Performance of ALS in Heterogeneous Environments

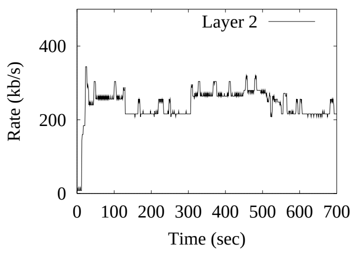
For testing the performance of ALS in heterogeneous network environments we repeated the simulations of Sec. 6.4.2 with a sender multicasting data to a set of receivers connected over links with different bottlenecks to the sender, see Fig. 6.5. The round trip propagation delay was set to 0.4 seconds and the maximum queuing delay of the routers was set to 0.15 seconds. Each link was shared between the ALS flow and 9 TCP connections that had the same end-to-end propagation delay as the ALS sender and receivers. The simulation was run for 800 seconds with the receiver connected to R1 joining the session after 300 seconds and leaving it again after 200 seconds.



(a) Layer 0



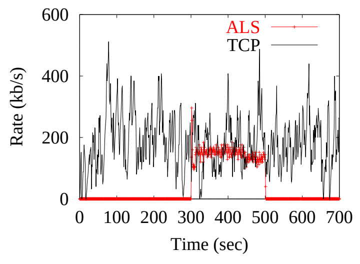
(b) Layer 1



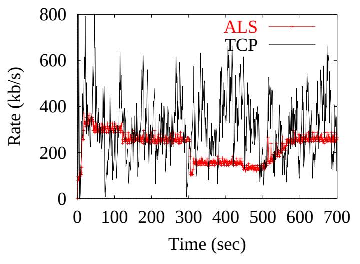
(c) Layer 2

Figure 7.8: Bandwidth distribution of the layers with ALS

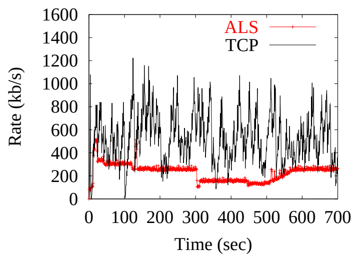
For distributing the bandwidth among the different layers we used the same approach as described in Sec. 6.2.1.1. That is, the sender determined the minimum (r_{min}) and maximum (r_{max}) reported rates and set the rate for the basic layer to r_{min}. The enhancement layers were then set to ($\frac{(\alpha \times r_{\text{max}}) - r_{\text{min}}}{K-1}$), with ($\alpha : \alpha \leq 1$) as a dampening factor and K as the number of layers to use. In this part, we set K to three, i.e., the data was distributed over three layers.



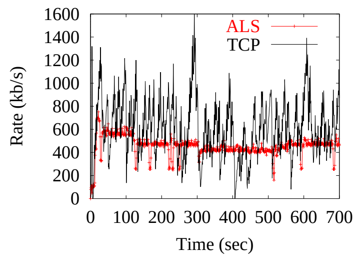
(a) Router 1



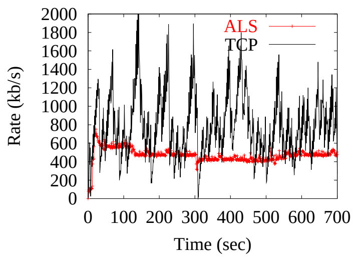
(b) Router 2



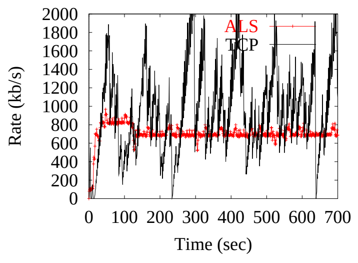
(c) Router 3



(d) Router 4



(e) Router 5



(f) Router 6

Figure 7.9: Bandwidth distribution between TCP and ALS at the different routers

The bandwidth distribution among the different layers reflects the heterogeneity of the receivers, see Fig. 7.8. During the first 300 seconds, the minimum rate is set to the bandwidth share of the receivers connected to R2. When a receiver connected to R1 joins the multicast group, the rate of the lowest layer is reduced to fit this new member. To compensate this reduction the rate of the upper layers is increased. This results then in a stable bandwidth share for the receivers listening to more than one layer, e.g., receivers connected to R6. Those receivers are hence not disturbed by the join and leave actions and the resulting changes in the bandwidth distribution. After the receiver connected to R1 has left the session, the rate of the lower layer is increased again to provide the receivers listening only to this layer with the best possible bandwidth share.

Compared to the results achieved in Sec. 6.4.2 we achieve with ALS a much more stable adaptation behavior. All receivers join a number of layers that fits on the links connecting them to the sender. For example, the receiver connected to R6 manages to join all the layers during the entire time of the simulation, whereas the receivers connected to R5 and R4 join only the first two layers.

7.5 Summary

To avoid having to probe the network in order to estimate the availability of resources, this chapter described a scheme called the adaptive load service (ALS) with which the network routers assist the end systems in determining the appropriate bandwidth share to use. ALS is to a large extent similar to MLDA. However, while with MLDA the TCP-friendly bandwidth share is estimated using an end-to-end control algorithm such as LDA, with ALS the bandwidth share of a flow is determined by the network routers. While ALS allows for simpler end systems it introduces additional overhead at the routers which need to maintain state information and process control information. Compared to MLDA this added complexity results in improved perceived QoS due to the increased stability of the bandwidth distribution as well as fulfilling the requirements of the max-min fairness criterion.

Chapter 8

Summary and Future Work

In this work, we studied the issue of providing TCP-friendly congestion control for multimedia communication. For exchanging control information between the end systems we relied in our work primarily on the real time transport protocol (RTP) instead of introducing a new control protocol.

In chapter 4, we first addressed the issue of TCP-friendly congestion control for unicast communication. In this context, we presented a new algorithm called the loss-delay based adaptation algorithm (LDA). LDA uses loss, delay and bottleneck bandwidth estimations to dynamically adjust the transmission behavior of the sender. Conducted simulations and measurements suggest LDA to be TCP-friendly and efficient in terms of loss avoidance and utilization over a wide range of tested parameters and to have a comparable performance to other schemes presented in the literature. The achieved results suggest also that RTP is in general suitable for exchanging control information between the sender and the receiver despite of the infrequency of the control messages. To accommodate a wider range of network parameters, the granularity of the loss representation in RTP should, however, be improved.

In chapter 5, we presented a framework for integrating congestion control with possible constraints imposed by the compression algorithm or the communication scenario on the one hand and maintaining a TCP-friendly bandwidth distribution on the other. Integrating this framework with LDA resulted then is a smoother adaptation behavior with less oscillations and at same the time did not alter the TCP-friendly character of LDA.

In chapter 6, a general framework for congestion control for heterogeneous multicast environments called MLDA was presented. MLDA integrates various techniques for realizing scalable and efficient congestion control such as layered data transmission and receiver-based rate estimations. We further presented different schemes for realizing scalable feedback mechanisms and round trip delay estimation. Additionally, with MLDA the layered transmission approach is further enhanced to allow for dynamically setting the number and sizes of the used layers to better account for the heterogeneity of the receivers. Conducted simulations suggest the efficiency of MLDA both in terms of satisfying the needs of heterogeneous receivers and TCP-friendly congestion control.

Finally, in chapter 7 we investigated the possible benefits of taking the adaptation decision not only based on loss and delay information collected by the end systems but also by using information generated by the network nodes about the resource availability. The results achieved in this part of the work were very promising in terms of achieved TCP-friendliness and stability of the adaptation process. However, in a real environment the processing overhead of the RTCP messages at the routers needs to be further investigated. In addition to the processing overhead at the routers, the proposed rate allocation scheme for ALS requires per flow state information. This might impose some problems in terms of scalability for supporting a large number of flows. Hence, further research is needed on bandwidth allocation algorithms that avoid the need for counting the number of active flows and the need for maintaining per-flow state information. One such an approach might be to set the bandwidth share given to a flow based on pricing or authentication information carried in the control packets.

Throughout this study we assumed cooperative end systems that deploy congestion control schemes. While this was and still is a realistic assumption it is doubtful if future applications will show such a social behavior. Such a development would then lead to the starvation of adaptive flows. To avoid such a situation future studies on congestion control should not only investigate efficient and fair end-to-end schemes but also look at possibilities for throttling greedy flows and protecting adaptive ones. This would probably require additional involvement of the network routers in the congestion control process in order to distinguish between greedy and adaptive flows [53].

While in this work we discussed the efficiency of the presented algorithms and frameworks over a variety of network topologies with different parameters we always assumed a wired network in which losses are only generated by network overload. In a more general approach we need to further consider the effects of losses caused by physical links such as wireless links. In this context, the congestion control schemes need to distinguish between losses caused temporarily by a sudden interruption at the physical layer and actual congestion. Differentiating between congestion and link caused losses when only observing one or a few packet losses during a short period of time is rather difficult. Actually, this is one of the main problems when using TCP over a wireless link. TCP reacts to the loss of each packet by reducing its transmission window. As the sender can not easily distinguish between congestion and link caused losses the sender might reduce its rate due to link losses as well. The same argumentation applies to congestion control schemes such as [146] that adjust the transmission rate of the sender after each loss notification. When observing the losses of a data stream over a time period of a few seconds or more one could use the distribution of the data losses as an indication of congestion or link caused losses. As with LDA the adaptation decision is already based on the loss observation over a period of a few seconds, the issue of loss distribution could be taken into account as well. The exact characteristics for distinguishing between congestion and link caused losses need still to be further investigated. Additionally, the reaction of a congestion control scheme such as LDA to link caused losses needs also to be further specified and tested.

To reduce the amount of observed losses by the receivers, it would be beneficial to integrate forward error correction mechanisms with adaptive congestion control. In this case, the sender would adjust the amount of transmitted redundant data based on the loss reports from the receiver and the TCP-friendly bandwidth share it could use. In this context, the sender needs to establish an equilibrium state between the amount of its

TCP-friendly bandwidth share dedicated to redundant data and the amount dedicated to new content in order to achieve the best tradeoff between low loss and a high content rate.

Further, in order to guarantee a minimal QoS level for multimedia flows, adaptation schemes need to be integrated with light-weight mechanisms for QoS provisioning. In such a scenario, the end systems might adapt the amount of their reserved resources based on the temporal prices or availability of those resources. In another scenario, a QoS class such as the differentiated services assured service class [67] might only provide proportional guarantees, i.e., it only guarantees that packets with a better mark will receive a better service than packets with worse markings. In this case, an adaptive sender might not only adapt the number of data packets it is sending based on the congestion situation in the network but also adjust the type of markings it is using. That is, when losses are observed in the used current level, the sender might refer to marking his packets with the code of a higher level if it is possible and possibly send a lower number of packets as well.

Appendix A

Investigation of ABR Switch Mechanisms

While lots of effort was done on behalf of the ATM Forum and the ITU standardization bodies in specifying the user network interface (UNI) for ATM's available bit rate (ABR) service the specification of the switch mechanisms was left to the manufacturers. In this part of the work, some of the algorithms proposed for dynamically distributing the available bandwidth at the switches such as CAPC, the OSU scheme and a modified version of the MIT proposal which were largely presented at the ATM Forum are compared. In our investigations, we will concentrate mainly on the fairness behavior of the schemes, link utilization, warm up periods and reaction to dynamically changing traffic conditions.

A.1 Introduction

Many applications, mainly handling data transfer, have the ability to reduce their sending rate if the network requires them to do so. Likewise, they may wish to increase their sending rate if there is extra bandwidth available within the network. This kind of applications is supported by an ATM layer service called the available bit rate service (ABR). Applications using the ABR service can expect the following quality of service commitments from the network:

1. The available bandwidth is fairly distributed among all active ABR connections.
2. A minimum cell rate that is agreed upon during the connection establishment phase.
3. Only a preset fraction of the sent cells can be dropped as long as the sending behavior of the application conforms to the negotiated values.

To provide these quality of service requirements appropriate traffic control mechanisms have to be used.

In its September 1994 meeting the ATM Forum voted for a rate based congestion control mechanism to be used in ATM networks. With the chosen algorithm the ABR sources must send a so called resource management (RM) cell every *Nrm* sent data cells. These cells indicate the current cell rate (CCR) and the desired one of the senders. The destination end system turns the RM cells around and sends them back to the source. Based on the traffic situation in the network the intermediate switches can determine the fair bandwidth shares the ABR connections should use in order to avoid congestion. These values are then written into the backward RM cells in the explicit rate (ER) field. The source should then increase or decrease its rate in accordance with the explicit rate noted in the RM cells.

In this part of the work different switch algorithms are described and compared with each other. In Sec. A.2 the different schemes are briefly introduced. Sec. A.3 describes the used simulation environment. In Sec. A.4, we explain the criterion used to describe the fairness of the separate schemes. Finally, in Sec A.5 the performance of the different schemes under various traffic conditions is simulated and examined.

A.2 Congestion Control and Avoidance in ATM Networks

In this study, various switch mechanisms are investigated and compared with each other. The first one, the MIT scheme, aims at estimating the fair bandwidth shares of the active connections based upon the information in the RM cells. The other schemes are more of congestion avoidance schemes that aim at driving the operation of the network towards the knee of the delay curve [145]. That is, these schemes aim at achieving an optimal overall performance with high bandwidth utilization and low buffer usage and delay.

A.2.1 The MIT Scheme

The original MIT scheme was proposed by Charny [32] in her master thesis at the Massachusetts Institute of Technology. With this scheme, the fair share is computed using an iterative procedure. Initially, the fair share is set to the link bandwidth divided by the number of active virtual connections (VC). All VCs, whose rates are less than the fair share are called "underloading VCs". If the number of underloading VCs increases at any iteration, the fair share is recomputed as follows:

$$F = \frac{R - \sum \text{Bandwidth of underloading VCs}}{\text{Number of VCs} - \text{Number of underloading VCs}} \tag{A.1}$$

with R as the total capacity available for the ABR service.

This procedure is repeated until a steady state is reached in which the fair share and the number of underloading VCs does not change [81]. In her work [32], Charny shows that two iterations are usually sufficient enough to reach convergence.

While this proposal fulfills the fairness requirements of the switch mechanisms it requires for the fair share computation $\theta(n)$ operations with n as the number of VCs for each received RM cell. For ATM switches supporting thousands of VCs the computational overhead would become extremely large. Kalampoukas et al. [89] present therefore an allocation algorithm that is based on the one introduced by Anna Charny but requires only $\theta(1)$ operations. This is reached by keeping track of the available bandwidth for the ABR connections. For more detailed description of the required computations see [89].

A.2.2 The OSU Congestion Avoidance Scheme

This scheme was developed by Jain et al. [83] at the Ohio State University (OSU). In this scheme, the switches measure their input rates over a fixed averaging interval of length T_m that is counted in cells [84]. Thereby, the input rate (R_in) is calculated as follows:

$$R_\mathrm{in} = \frac{\text{number of received cells}}{T_\mathrm{m}} \tag{A.2}$$

With this calculated input rate a load factor (L_f) can be computed which can then be used to indicate the over- or underload state of the switch.

$$L_\mathrm{f} = \frac{R_\mathrm{in}}{R_\mathrm{T}} \tag{A.3}$$

Here, the target rate (R_T) indicates the rate level at which an optimal network performance can be achieved with high utilization and low buffer usage. The target rate (R_T) is usually set to a value between 85% and 95% of the bandwidth available for the ABR service. During the steady state the utilization should be as high as the target rate and during the transient state with lots of bursty connections the utilization could be even higher.

The sources are then asked to change their current sending rate inversely proportional to the calculated load factor. That is, with a load factor of 0.5 for example the sources should double their sending rates to avoid an underload situation at the switch.

To assure fairness and a minimal cell rate the active connections (N) must at least receive the minimum fair share that is calculated as follows:

$$F = \frac{R_\mathrm{T}}{N} \tag{A.4}$$

A.2.3 Congestion Avoidance using Proportional Control (CAPC)

Basically, this algorithm developed by Andy Barnhart from Hughes Systems [10] shows a great similarity to the OSU scheme described in Sec. A.2.2. Just as in the OSU scheme the switches set the target utilization slightly below 1 and compute the input rate over averaging intervals. With the computed input rate a load factor (z) can then be calculated. The main difference to the OSU scheme is in the way the explicit rates are determined.

During underload states ($z < 1$) the fair share (F)of the ABR connections is increased as follows:

$$F = F \times \min(I_\mathrm{max}, 1 + (1 - z) * S_\mathrm{up}) \tag{A.5}$$

Here, S_up is a slope parameter in the range of 0.025 to 0.1 and I_max is the maximum increase allowed and is usually set to 1.5.

During overload states ($z > 1$) the fair share of the ABR connections is decreased as follows:

$$F = F \times \max(D_\mathrm{max}, 1 + (z - 1) * S_\mathrm{down}) \tag{A.6}$$

Here, S_{down} is a slope parameter in the range of 0.2 to 0.8 and $D_{\max}$ is the maximum decrease allowed and is usually set to 0.5.

In addition to the load factor the scheme uses a queue threshold. Whenever the queue length is over this threshold, a congestion indication (CI) bit is set in all RM cells. This prevents all the sources from increasing their rates and allows the queue to drain out more quickly.

During the steady state with $(z = 1)$ this scheme achieves an oscillation-free steady state. As the frequency of the oscillation is a function of $(1 - z)$ having a load factor of 1 yields an infinite oscillation period.

A.3 Simulation Environment

For testing the performance of the different schemes we used a chain topology with five switches and 21 connections originating from seven sources, whereas each source was the origin of three similar connections. The distance between two neighboring switches was 1000 km and the distance from a source to a switch was 0.4 km. All the links had a bandwidth of 45 Mb/s and the peak cell rate (PCR) of all connections was set to the link bandwidth. Depending on the number of switches they have to pass, three kinds of traffic could be distinguished:

- Long distance connections: Connections starting from source 0 traverse all four links. These connections will be denoted as VC_0.

- Medium distance connections: Connections starting from sources 1 and 4 traverse two links. These connections will be denoted as VC_1 and VC_4.

- Short distance connections: Connections starting from sources 2, 3, 5 and 6 traverse only one link. Theses connections will be denoted as VC_2, VC_3, VC_5 and VC_6.

For the fairness investigations we used persistent sources. These are greedy sources that can always send data with the maximum allowed cell rate and impose thereby the most stringent constraints on the network.

In the second stage of our performance tests, we investigated the behavior of the schemes in the presence of dynamically changing traffic conditions. The same model as in the first step was used here as well, but with the connections starting from source 0 being shut on and off to introduce load changes in the network. The time required to achieve a stable bandwidth distribution and the changes in the buffer length at the switches as well as the fairness of the schemes were investigated in this case.

A.4 Fairness Testing Criterion

As a fairness criterion we used the max-min fairness definition as was described in the ATM Forum traffic management specification 4.0 [162]. With this definition a connection's

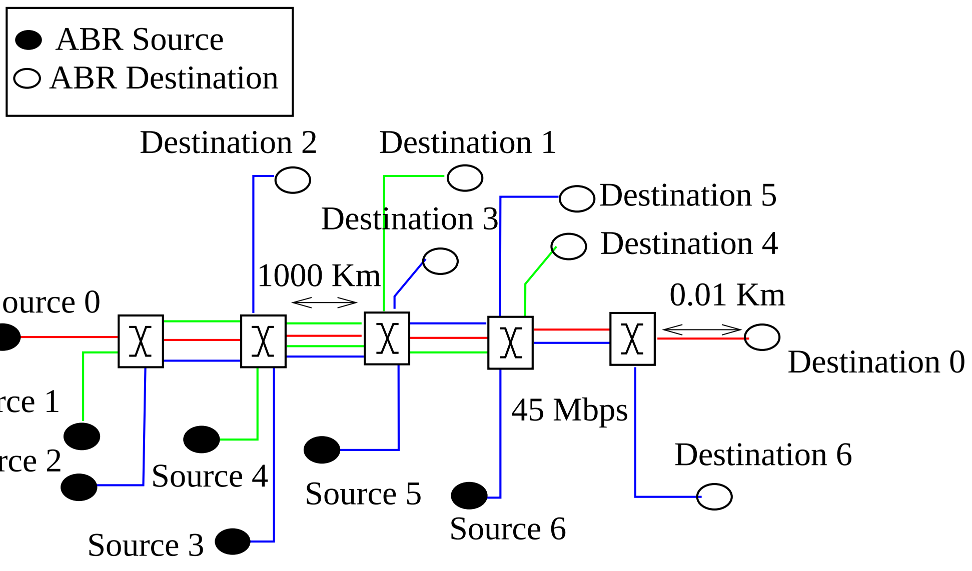


Figure A.1: ABR test topology

bandwidth request is either satisfied by the switch or set to the fair share (F) calculated by the switch. This fair share is determined as follows:

$$F = \frac{\text{B} - \sum \text{Bandwidth of constrained connections}}{\text{Number of connections} - \text{Number of constrained connections}} \tag{A.7}$$

with B as the bandwidth dedicated to the ABR service and a connection is considered to be constrained if the rate it is currently using is smaller than the fair bandwidth share calculated at this switch.

Using the max-min criterion to determine the appropriate bandwidth distribution we would get the distribution presented in Tab. A.1. For the fairness estimation we used the fairness index suggested by Jain [80] and defined as follows:

$$F = \frac{\left(\sum x_i\right)^2}{n \sum x_i^2} \quad \begin{array}{ll} \text{with } x_i & = \text{ratio of the actual throughput} \\ & \quad \text{to the fair throughput} \\ \text{and } n & = \text{the number of connections} \end{array} \tag{A.8}$$

VC_0	VC_1	VC_2	VC_3	VC_4	VC_5	VC_6
3.75	3.75	7.5	3.75	3.75	7.5	11.25

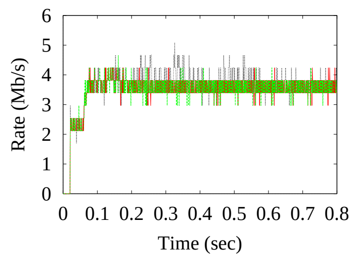
Table A.1: Expected fair bandwidth share for the single VCs in Mb/s

A.5 Performance Results

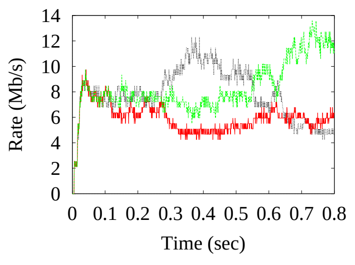
Using the simulation environment presented in Sec. A.3 and the fairness criterion of Sec. A.4, we tested the behavior of the different algorithms using persistent and dynamically changing traffic sources. To introduce the dynamic traffic changes we shut the traffic originating from source 0 on and off in intervals of 0.1 seconds.

A.5.1 The OSU Scheme

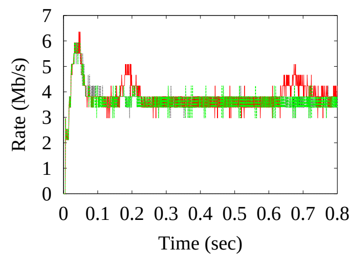
Here, we used a target rate of 95% of the available link bandwidth and would therefore expect to achieve a bandwidth distribution of only 95% of the values depicted in Tab. A.1.



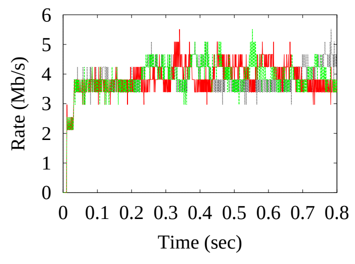
(a) Source 0



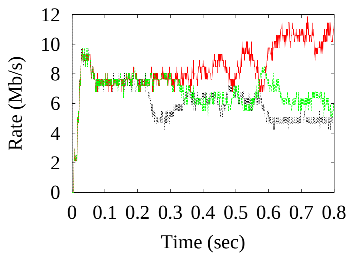
(b) Source 2



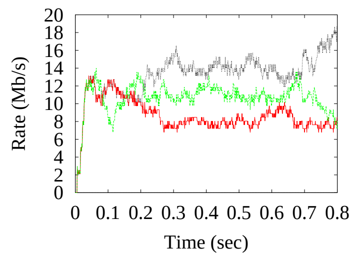
(c) Source 3



(d) Source 4



(e) Source 5



(f) Source 6

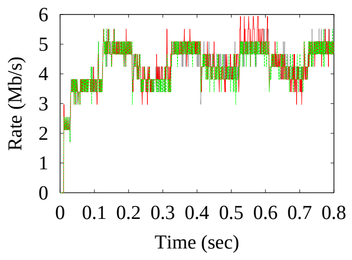
Figure A.2: Performance of the OSU scheme under persistent load

The results achieved for the OSU scheme depicted in Tab. A.2 reveal some serious drawbacks of the algorithm. While the average rate achieved for a complete path, i.e., the average rate achieved by the three connections traversing the same path, is very close to our expectations, the values of the single VCs vary strongly. This is of course surprising as all of the VCs traversing the same path are identical. Investigating the behavior of the switches more closely revealed the following: the switches update the explicit rate of all RM cells that arrive during the same measurement period by the same load factor. So, if it happens that during a measurement interval RM cells from only a part of the VCs traversing a common path arrived at the switch the rate of only these connections will be adjusted with the current load factor. The rate of the other connections traversing this path will only be updated in one of the next measurement intervals with another load factor. This results then in different allowed rates at the sources.

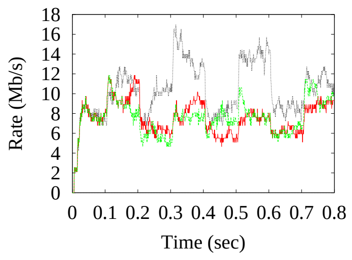
Another problem with the OSU scheme is its unfairness towards short distance traffic during overload periods. We have mentioned that the switches write the fair rate share in the backward RM cells. While this ensures that the rate is updated using the most current load factor it leads to some unfairness in the case of connections with different round trip times. As all connections contribute to the switch congestion in the same way we would expect the switch to reduce the rate of all connections using the same overload factor.

Connection	VC_0	VC_1	VC_2	VC_3	VC_4	VC_5	VC_6	F
1	3.63	3.57	7.5	3.65	3.58	6.06	8.8	
2	3.58	3.59	7.67	3.68	3.58	6.1	11.42	
3	3.60	3.57	6.2	3.66	3.59	9.22	11.92	
average	3.60	3.57	7.12	3.66	3.58	7.138	10.71	0.99

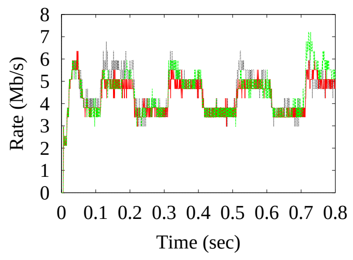
Table A.2: Achieved bandwidth in Mb/s for the single VCs using the OSU scheme



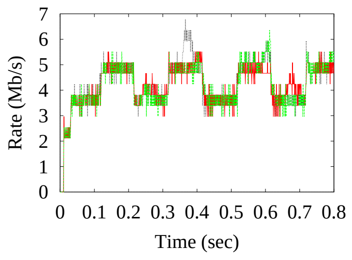
(a) Source 1



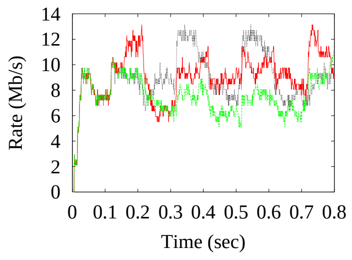
(b) Source 2



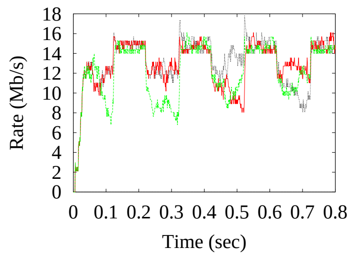
(c) Source 3



(d) Source 4



(e) Source 5



(f) Source 6

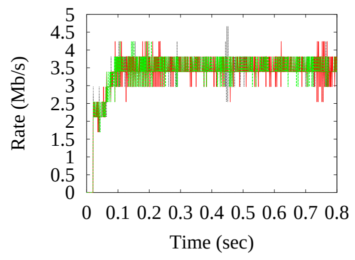
Figure A.3: Performance of the OSU scheme under dynamically changing load

However, due to the short round trip times of the short distance connections their RM cells arrive back at the switch much earlier than those of the long distance connections. Thereby, these sources will be forced to reduce their bandwidth with a higher overload factor. As the short distance connections reduce their rates the load factor will be reduced as well. With the smaller load factor the explicit rates of the later arriving RM cells of the long distance connections will be less reduced than was the case for the short distance traffic. In short, this means that the long distance connections can benefit longer from a high cell rate and utilize a bandwidth share larger than their fair share. This has also the negative effect of lengthening the wrap up period. On the other hand, during underload phases the short distance connections can increase their rates faster with a smaller overload factor. That is, in this situation the short distance connections can benefit from the small factor, increase their rate and thereby increase the load factor. When the RM cells of the long distance connections arrive after the longer round trip delay their rates can only be changed by the larger factor and the rate of the long distance connections can thereby only be increased slower than the rate changes of the short distance ones.

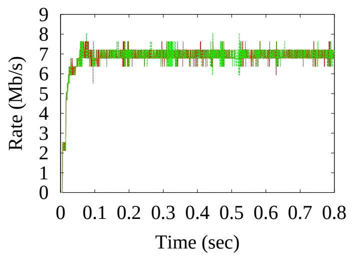
Fig. A.3 depicts the behavior of the OSU scheme under dynamic load patterns. With the VCs originating from source 0 sending data only periodically, the OSU scheme displays the same behavior as in Tab. A.2 with an uneven bandwidth distribution among the VCs belonging to the same path.

A.5.2 The CAPC Scheme

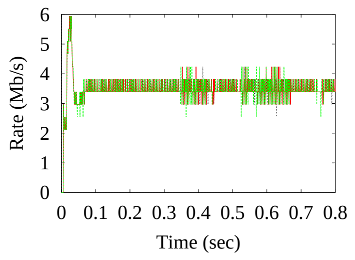
With all of the sources starting with an initial cell rate (ICR) of only $\frac{1}{20}$ of the peak cell rate, CAPC results in an ideal behavior. All the VCs traversing a common path receive a similar bandwidth share, the bandwidth is fairly distributed and we can only notice a small transient period.



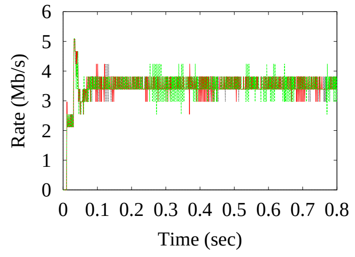
(a) Source 0



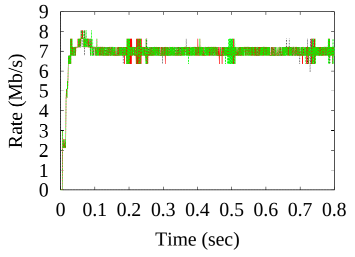
(b) Source 2



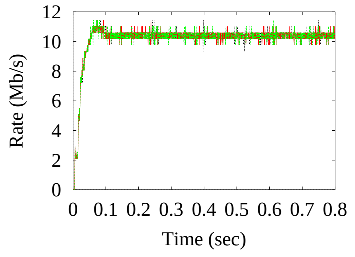
(c) Source 3



(d) Source 4



(e) Source 5



(f) Source 6

Figure A.4: Performance of the CAPC scheme under persistent load

For the case of an initial cell rate as high as the peak cell rate we notice, however, a bug in the algorithm as it was described in [10]. Due to the long lasting congestion at the switches caused by the high initial cell rate the fair share value is reduced several times until it is finally represented by 0. In this case, the fair share value will remain 0 even if all the sources reduced their rates. Fig. A.5 shows the results achieved for sources 0, 1 and 2. A possible solution to this problem is to add a minimum threshold value for the fair share. This minimum threshold could be set as the available bandwidth (B) divided by the number of connections seen in the last measurement interval (N). This might

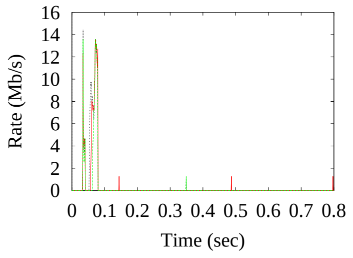
Connection	VC_0	VC_1	VC_2	VC_3	VC_4	VC_5	VC_6	F
1	3.5	3.6	7.2	3.6	3.6	7.2	10.8	
2	3.6	3.6	7.2	3.6	3.6	7.2	10.7	
3	3.5	3.6	7.2	3.6	3.6	7.2	10.8	
average	3.5	3.6	7.2	3.6	3.6	7.2	10.8	0.99

Table A.3: Achieved bandwidth share in Mb/s for the single VCs using CAPC

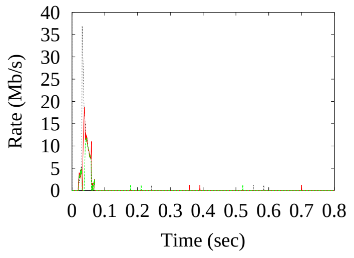
be combined with the setting of the congestion indication (CI) bit to force a rapid rate reduction at the sources. With this enhancement the switch would set the fair rate share during overload periods as follows:

$$F = \max(F \times \max(D_{\min}, 1 + (z-1) \times S_{\text{down}}), \frac{B}{N}) \tag{A.9}$$

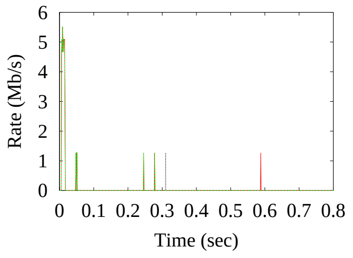
Here, S_{down} is a slope parameter in the range of 0.2 to 0.8 and $D_{\max}$ is the maximum decrease allowed and is usually set to 0.5.



(a) Source 0



(b) Source 1

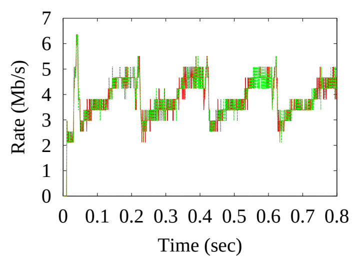


(c) Source 2

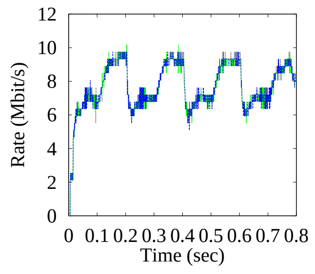
Figure A.5: Performance of the CAPC scheme under persistent traffic load with all the sources starting with a peak cell rate

As the CAPC scheme updates the received RM cells each measurement interval with the newly calculated load factor the same unfairness noticed with the OSU scheme for the case of connections with different round trip times applies here as well.

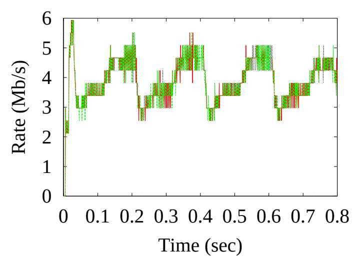
The simulations run for the dynamically changing load case shown in Fig. A.6 reveal that the scheme is capable of reaching a stable state after each load change in a few round trip times. Also, unlike the OSU scheme all VCs belonging to the same source receive the same bandwidth share. Note, however, that while the scheme can reduce its rate in about only one round trip time when new traffic is added to the switches it requires a few round trip times until it can utilize the additional bandwidth.



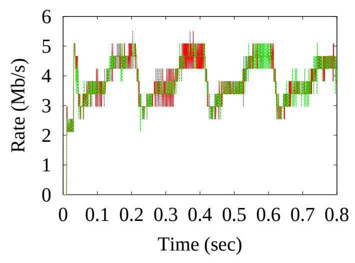
(a) Source 1



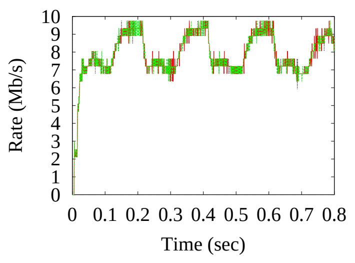
(b) Source 2



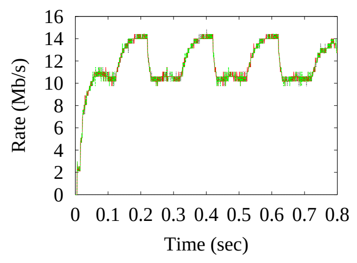
(c) Source 3



(d) Source 4



(e) Source 5



(f) Source 6

Figure A.6: Performance of the CAPC scheme under dynamically changing load

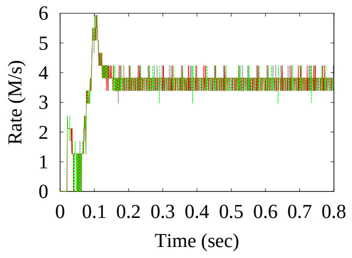
A.5.3 The MIT Scheme

Even though the MIT based scheme that was used here requires only $\theta(1)$ operations per received RM cell it is still more complicated than the previous algorithms described in sections A.2.2 and A.2.3.

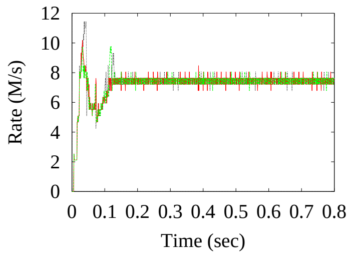
However, this added complexity results in fair bandwidth distribution, a stable steady state and high utilization, see Fig. A.7 and Tab. A.4. Also, while this scheme requires more arithmetical operations most of the needed operations are simple addition and comparison operations that do not cause a high computational overhead.

This scheme solves the unfairness problems noticed in the OSU and CAPC schemes during the transient periods. The rates are not updated proportional to the CCR value noted in the RM cells. Instead, the switches write the calculated fair share directly in the RM cells. So, as soon as the switch has enough information about the connections and can determine the appropriate fair share all the connections will be updated in the same way.

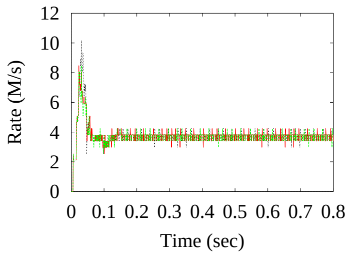
Under dynamically changing traffic conditions we can watch that the MIT scheme starts oscillating and does not reach a stable state during the on and off phases, see Fig. A.8. This is actually caused by a severe drawback of the algorithm that only gets obvious through changing the traffic conditions. As Fig A.9 reveals, in the case of the persistent traffic simulation the buffer requirements of this scheme increase at the beginning when all the sources start sending and then reaches a constant level that is maintained throughout the simulation run. As the scheme's fair share calculation is based on a 100%



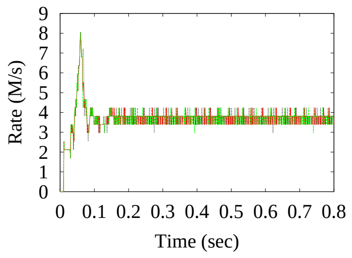
(a) Source 1



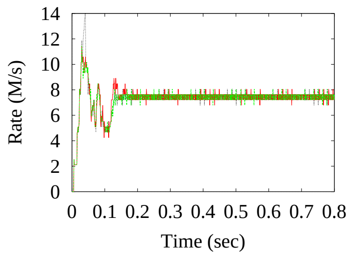
(b) Source 2



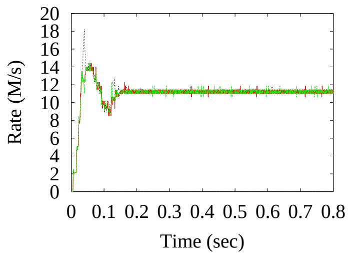
(c) Source 3



(d) Source 4



(e) Source 5



(f) Source 6

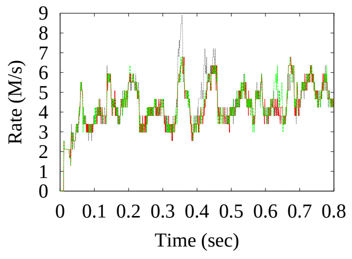
Figure A.7: Performance of the MIT scheme under persistent load

Connection	VC_0	VC_1	VC_2	VC_3	VC_4	VC_5	VC_6	F
1	3.4	3.6	7.3	3.8	3.7	7.5	11.2	
2	3.4	3.6	7.3	3.8	3.7	7.4	11.2	
3	3.4	3.6	7.3	3.8	3.7	7.5	11.2	
average	3.4	3.6	7.3	3.9	3.7	7.5	11.2	0.99

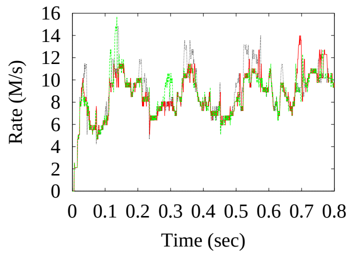
Table A.4: Achieved bandwidth share in Mb/s for the single VCs using the MIT scheme

link utilization and the fair share is reached rather fast, the buffer has no time to drain out.

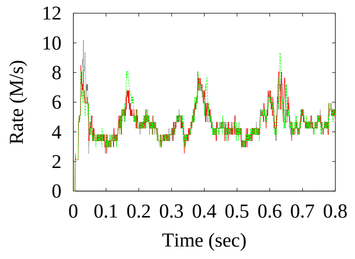
In the second case, with the traffic of source 0 being shut on and off, the buffer is increased at the beginning of each on period. During the off periods it drains out for a while until the all the remaining sources reach the fair share again. As the scheme calculates the fair share rather fast, the number of cells that can drain out after entering an off state is less than the number of cells that get accumulated at the buffer at the beginning of each on period. This results in the gradual increase of the buffer at the switches. A simple method of avoiding this problem would be to calculate the fair share not only based on the RM cells but based on the buffer length as well.



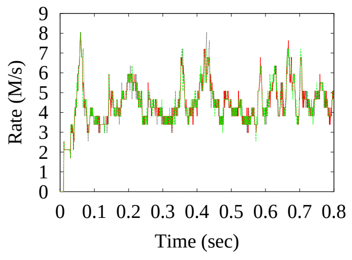
(a) Source 1



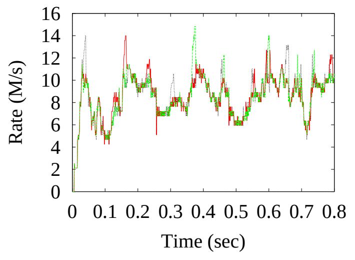
(b) Source 2



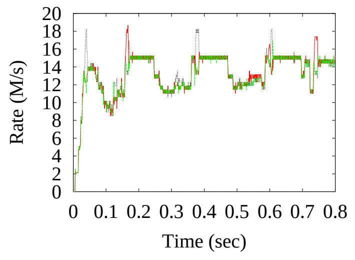
(c) Source 3



(d) Source 4



(e) Source 5



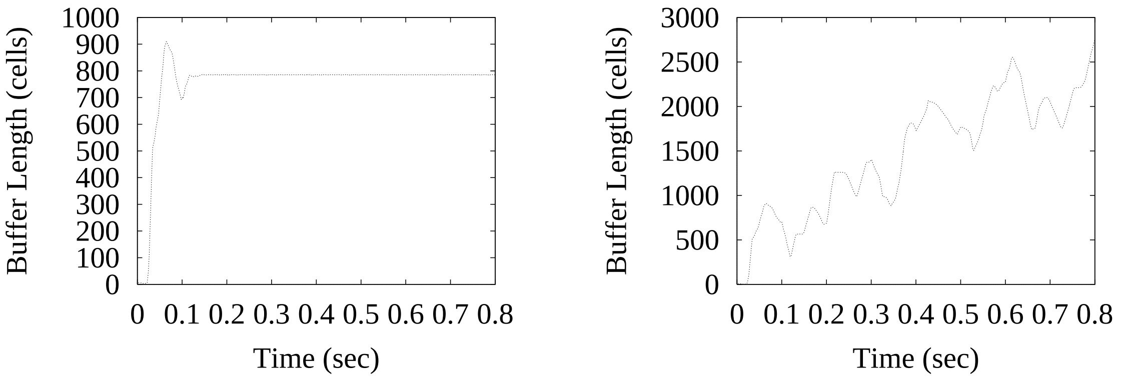
(f) Source 6

Figure A.8: Performance of the MIT scheme under dynamically changing load

A.6 Summary

In this part of the work, we mainly concentrated on the aspects of fairness, wrap up time and utilization in the presence of persistent and dynamic traffic. The obtained results suggest mostly the superiority of the more complicated schemes such as the MIT scheme. The OSU scheme shows some unfairness towards connections originating from the same source. The CAPC scheme shows stable behavior but is slow in reacting to underload situations.

Algorithms that are based on the contents of the RM cells such as the MIT scheme are more straight forward schemes. They obtain the needed information from the RM cells and do not need to monitor the traffic or number of VCs. While this results in rather more complicated schemes it allows for more stable performance that does not depend on the chosen averaging parameters or measurement intervals. However, as such schemes do not monitor the buffer length there might be some problems with buffer management as we have seen in Sec. A.5.3. Also, changes in the bandwidth available for the ABR service have to be announced by some other entities in the switch as such schemes can not monitor this by themselves. In terms of wrap up time, i.e., the time required for achieving a stable bandwidth distribution, we can observe that schemes such as OSU and CAPC which measure the incoming rate and available bandwidth have shorter transient periods than MIT for example. As MIT bases its fair rate estimations on the control messages, it requires a few rounds of messages to determine a stable bandwidth distribution. Measuring the number of incoming flows and packets allows OSU and CAPC to estimate the



(a) Persistent traffic (b) Dynamic traffic

Figure A.9: Buffer requirements of the first switch with the MIT scheme

fair shares after one or a few measurement periods only. Hence, a combination between the two approaches would most probably result in more stable schemes that can benefit from the merits of both approaches.

Bibliography

[1] The emerging digital economy, Apr. 1998. Available from http://www.ecommerce.gov/emerging.html.

[2] I. Adiri and B. Avi-Itzhak. Queueing models for time-sharing service systems. In *Proc. of the 5th International Conference on Operations Research*, pages 205–222, Venice, Italy, 1969.

[3] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Int. Conf. VLDB*, Santiago, Chile, Sept. 1994.

[4] C. Albuquerque, B. Vickers, and T. Suda. An end-to-end source-adaptive multi-layered multicast (SAMM) algorithm. In *9th International Packet Video Workshop*, New York, USA, Apr. 1999.

[5] W. Almesberger, T. Ferrari, and J. L. Boudec. SRP:scalable resource reservation for the internet. In *IWQoS'98*, Napa, California, May 1998.

[6] E. Amir, S. McCanne, and R. Katz. Receiver-driven bandwdith adaptation for light-weight sessions. In *ACM Multimedia'97*, Seattle, Nov. 1997.

[7] E. Amir, S. McCanne, and H. Zhang. An application level video gateway. In *Proc. of ACM Multimedia*, San Francisco, California, Nov. 1995.

[8] A. T. Aurrecoechen, Cristina Campbell and L. Hauw. A survey of QoS architectures. *Multimedia Systems Journal*, 6(3):138–151, May 1998. Special Issue on QoS Architectures.

[9] H. Balakrishnan, S. Seshan, E. Amir, and R. H. Katz. Improving TCP/IP performance over wireless networks. In *Proc. of 1st ACM Conference on Mobile Computing and Networking*, Berkeley, California, Nov. 1995.

[10] A. W. Barnhart. Explicit rate performance evaluations. Technical Report 94-0983, ATM Forum, Oct. 1994.

[11] S. M. Bellovin and W. R. Cheswick. Network firewalls. *IEEE Communications Magazine*, 32(9):50–57, Sept. 1994.

[12] L. Berger, D. Gan, G. Swallow, P. Pan, F. Tommasi, and S. Moledin. Rsvp refresh reduction extensions. Internet Draft, Internet Engineering Task Force, June 1999. Work in progress.

[13] S. Bhattacharyya, D. Towsley, and J. Kurose. The loss path multiplicity problem for multicast congestion control. In *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, New York, Mar. 1999.

[14] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated service. Request for Comments (Informational) 2475, Internet Engineering Task Force, Dec. 1998.

[15] J.-C. Bolot. End-to-end packet delay and loss behavior in the Internet. In D. P. Sidhu, editor, *SIGCOMM Symposium on Communications Architectures and Protocols*, pages 289–298, San Francisco, California, Sept. 1993. ACM. also in *Computer Communication Review* 23 (4), Oct. 1992.

[16] J.-C. Bolot. Cost-quality tradeoffs in the Internet. *Computer Networks and ISDN Systems*, pages 645–651, 1996.

[17] J.-C. Bolot and A. U. Shankar. Dynamical behavior of rate-based flow control mechanisms. *ACM Computer Communication Review*, 20(2):35–49, Apr. 1990.

[18] J.-C. Bolot and T. Turletti. A rate control mechanism for packet video in the internet. In *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, Toronto, Canada, June 1994.

[19] J.-C. Bolot, T. Turletti, and I. Wakeman. Scalable feedback control for multicast video distribution in the internet. In *SIGCOMM Symposium on Communications Architectures and Protocols*, pages 58–67, London, England, Aug. 1994. ACM.

[20] J. M. Boyce and R. D. Gaglianello. Packet loss effects on MPEG video sent over the public internet. In *ACM Multimedia '98*, Bristol, UK, September 1998. Multimedia 98 Commitee, ACM.

[21] B. Braden and L. Zhang. Resource ReSerVation protocol (RSVP) – version 1 message processing rules. Request for Comments (Proposed Standard) 2209, Internet Engineering Task Force, Oct. 1997.

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

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

[24] L. Breslau and S. Shenker. Best-effort versus reservations: A simple comparative analysis. In *SIGCOMM'98*, vancouver, Canada, September 1998.

[25] I. Busse, B. Deffner, and H. Schulzrinne. Dynamic QoS control of multimedia applications based on RTP. *Computer Communications*, 19(1):49–58, Jan. 1996.

[26] A. Campbell. *A Quality of Service Architecture*. PhD thesis, Lancaster University, Lancaster, England, Jan. 1996.

[27] A. Campbell, D. Hutchinson, and C. Aurrecoechea. Dynamic QoS management for scalable video flows. In *Proc. International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, Lecture Notes in Computer Science, pages 107–118, Durham, New Hampshire, Apr. 1995. Springer.

[28] G. Carle and E. W. Biersack. Survey of error recovery techniques for IP-based audio-visual multicast applications. *IEEE Network Magazine*, November/December 1997.

[29] R. L. Carter and M. E. Crovella. Measuring bottleneck link speed in packet-switched networks. Technical Report BU-CS-96006, Computer Science Departement, Boston University, Mar. 1996.

[30] S. Casner and S. Deering. First IETF Internet audiocast. *ACM Computer Communication Review*, 22(3):92–97, July 1992.

[31] CCITT. H.261: Video codec for audiovisual services at px64 kbit/s. CCITT White book H.261, CCITT, 1990.

[32] A. Charny. An algorithm for rate allocation in a packet-switching network with feedback. Technical Report MIT/TR-601, MIT, Cambridge, Massachusetts, May 1994.

[33] R. Chipalkatti, J. F. Kurose, and D. Towsley. Scheduling policies for real-time and non-real time traffic in a statistical multiplexer. In *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, pages 774–783, Ottawa, Canada, Apr. 1989. IEEE.

[34] D. Chiu and R. Jain. Analysis of the increase/decrease algorithms for congestion avoidance in computer networks. *Journal of Computer Networks and ISDN*, 17(1):1–14, June 1989.

[35] K. Claffy and G. Miller. The natureof the beast: Recent traffic measurements from an Internet backbone. In *INET '98*, Palexpo Konferenz Zentrum, Genf, July 1998. Internet Society.

[36] K. G. Coffman and A. M. Odlyzko. The size and groth rate of the Internet, July 1998. Available from http://www.research.att.com/ãmo.

[37] D. E. Comer. *Internetworking with TCP/IP: Client-Server Programming and Applications, Windows Sockets Version*, volume 3. Prentice Hall, Upper Saddle River, New Jersey, 1997.

[38] A. Conta and S. Deering. Internet control message protocol (ICMPv6) for the internet protocol version 6 (ipv6). Technical Report RFC 1885, Internet Engineering Task Force, Dec. 1995.

[39] S. E. Deering. Host extensions for IP multicasting. Technical Report RFC 1112, Internet Engineering Task Force, Aug. 1989.

[40] S. E. Deering. *Multicast routing in a datagram internetwork.* PhD thesis, Stanford University, Palo Alto, California, Dec. 1991.

[41] D. DeLucia and K. Obraczka. A multicast congestion control mechanism for reliable multicast. Technical Report 97-685, University of South California, Computer Science Departement, Aug. 1999.

[42] W. Diot, Christophe Dabbous and J. Crowcroft. Multipoint communication: Asurvey of protocols, functions, and mechanisms. *Journal on Selected Areas in Communications*, 15(3):277– 290, April 1997.

[43] A. B. Downey. Using pathchar to estimate internet link charachteristics. In *Special Interest Group on Data Communication (SIGCOMM'99)*, Cambridge, USA, Aug. 1999.

[44] R. El-Marakby and D. Hutchison. Towards managed real-time communications in the Internet environment. In *The Fourth IEEE Workshop on the Architecture and Implementation of High Performance Communication Systems (HPCS'97)*, Chalkidiki, Greece, June 1997.

[45] A. Eleftheriadis. Mpeg-4. In *W3C Synchronization of Multimedia Meeting*, New York, New York, May 1997.

[46] F. Emanuel. Investigation of scalability and fairness of adaptive multimedia applications. Studienarbeit, School of Engineering, TU Berlin, Berlin, May 1997. Work in progress.

[47] H. Eriksson. MBone – the multicast backbone. In *Proceedings of the International Networking Conference (INET)*, pages CCC–1 – CCC–5, San Francisco, California, Aug. 1993. Internet Society.

[48] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, and L. Wei. Protocol independent multicast-sparse mode (PIM-SM): protocol specification. Technical Report RFC 2117, Internet Engineering Task Force, June 1997.

[49] W.-c. Feng, D. D. Kandlur, D. Saha, and K. G. Shin. Adaptive packet marking for providing differentiated services in the Internet. In *ICNP'98*, Austin, Texas, October 1998.

[50] D. Ferrari, A. Banerjea, and H. Zhang. Network support for multimedia: A discussion of the Tenet approach. *Computer Networks and ISDN Systems*, 10:1267–1280, July 1994.

[51] S. Floyd. Connections with multiple congested gateways in packet-switched networks, part 1–one-way traffic. *ACM Computer Communication Review*, 21(5), Oct. 1991.

[52] S. Floyd. TCP and explicit congestion notification. A PostScript version of this document is available from ftp://ftp.ee.lbl.gov/papers/tcp_ecn.4.ps.Z, 1994.

[53] S. Floyd and K. Fall. Promoting the use of end-to-end congestion control in the internet. *IEEE/ACM Transactions on Networking*, Aug. 1999.

[54] S. Floyd and V. Jacobson. On traffic phase effects in packet-switched gateways. *Internetworking: Research and Experience*, 3(3):115–156, Sept. 1992.

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

[56] S. Floyd and F. Kevin. Router mechanisms to support end-to-end congestion control. Technical report, LBL-Berkeley, Feb. 1997.

[57] S. Floyd and F. Kevin. Promoting the use of end-to-end congestion control in the Internet. Technical report, LBL-Berkeley, Feb. 1998. Under submission.

[58] H. J. Fowler and W. E. Leland. Local area network traffic characteristics, with implications for broadband network congestion management. *IEEE Journal on Selected Areas in Communications*, 9(7):1139–1149, Sept. 1991.

[59] Y. Gao and J. C. Hou. Racoom: A rate-based congestion control scheme for multicast. Note to the Internet Reliable Multicast Group mailing list, Mai 1999.

[60] M. Grossglauser, S. Keshav, and D. Tse. RCBR: A simple and efficient service for multiple time-scale traffic. *IEEE/ACM Transactions on Networking*, pages 741 – 755, Dec. 1997.

[61] M. Handely and S. Floyd. Strawman specification for TCP friendly reliable multicast congestion control, 1998. Note to the Internet Reliable Multicast Group mailing list.

[62] M. Handley. SAP: Session announcement protocol. Internet Draft, Internet Engineering Task Force, Nov. 1996. Work in progress.

[63] M. Handley, H. Schulzrinne, and E. Schooler. SIP: Session initiation protocol. Internet Draft, Internet Engineering Task Force, July 1997. Work in progress.

[64] M. Handley, H. Schulzrinne, and E. Schooler. SIP: session initiation protocol. Internet Draft, Internet Engineering Task Force, May 1998. Work in progress.

[65] B. G. Haskell, A. Puri, and A. N. Netravali. *Digital video: an introduction to MPEG-2*. Chapman & Hall, New York, New York, 1996.

[66] C. L. Hedrick. Routing information protocol. Technical Report RFC 1058, Internet Engineering Task Force, June 1988.

[67] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski. Assured forwarding PHB group. Request for Comments (Proposed Standard) 2597, Internet Engineering Task Force, June 1999.

[68] J. C. Hoe. Start-up dynamics of TCP's congestion control and avoidance schemes. Master's thesis, Department of Electrical Engineering and Computer Science, MIT, Cambridge, Massachusetts, May 1995.

[69] M. Hoffman, J. Nonnenmacher, J. Rosenberg, and H. Schulzrinne. A taxonomy of feedback for multicast. Internet Draft, Internet Engineering Task Force, June 1999. Work in progress.

[70] U. Horn. *Skalierbare Videocodierung mit einer bewegungskompensierten rtlich-zeitlichen Aufl sungspyramide in Verbindun mit E_8-Gitter-Quantisierung.* PhD thesis, Erlangen-N rnberg, Erlangen, Germany, Sept. 1997.

[71] U. Horn and B. Girod. Scalable video coding for the internet,. In *8th Joint European Networking Conference*, Edinburgh, England, May 1997.

[72] International Telecommunication Union. Visual telephone systems and equipment for local area networks which provide a non-guaranteed quality of service. Recommendation H.323, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, May 1996.

[73] ITU-T/CCITT. Interactive test methods for audiovisual communications. Recommendation P.920, International Telecommunication Union.

[74] S. Jacobs and A. Eleftheriadis. Providing video services over networks without quality of service guarantees. In *RTMW'96*, Sophia Antipolis, France, Oct. 1996.

[75] V. Jacobson. Congestion avoidance and control. *ACM Computer Communication Review*, 18(4):314–329, Aug. 1988. Proceedings of the Sigcomm '88 Symposium in Stanford, CA, August, 1988.

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

[77] J. Jaffe. Bottleneck flow control. *IEEE Transactions on Communications*, July 1980.

[78] R. Jain. A delay-based approach for congestion avoidance in interconnected heterogeneous computer networks. *ACM Computer Communication Review*, 19(5):56–71, Oct. 1989. also Digital Equipment Corporation Technical Report DEC-TR-566.

[79] R. Jain. *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling.* John Wiley, New York, 1991.

[80] R. Jain. Fairness: How to measure quantitatively? Technical Report 94-0881, ATM Forum, Sept. 1994.

[81] R. Jain. Congestion control and traffic management in ATM networks: Recent advances and a survey. *Computer Networks and ISDN Systems*, Feb. 1995.

[82] R. Jain, S. Fahmy, S. Kalyanaraman, and R. Goyal. ABR switch algorithm testing: A case study with ERICA. Technical Report 96-1267, ATM Forum, Oct. 1996.

[83] R. Jain, S. Kalyanaraman, and R. Viswanthan. The OSU scheme for congestion avoidance using explicit rate indication. Technical Report 94-883, ATM Forum, Sept. 1994.

[84] R. Jain, S. Kalyanaraman, R. Viswanthan, and R. Goyal. A sample switch algorithm. Technical Report 95-178R1, ATM Forum, Feb. 1995.

[85] S. Jamin, P. B. Danzig, S. Shenker, and L. Zhang. A measurement-based admission control algorithm for integrated services packet networks. In *SIGCOMM Symposium on Communications Architectures and Protocols*, pages –, Cambridge, Massachusetts, Sept. 1995.

[86] T. Jiang, E. Zegura, and M. Ammar. Inter-receiver fair multicast communication over the internet. In *Proc. International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, Basking Ridge, New Jersey, June 1999.

[87] K. Jonas, J. Moedeker, and M. Kretschmer. Get a KISS - communication infrastructure for streaming service in a heterogeneous environment. In *ACM Multimedia '98*, Bristol,UK, September 1998. Multimedia 98 Commitee, ACM.

[88] L. Kalampoukas and A. Varma. Two-way TCP traffic over ATM: Effects and analysis. In *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, Kobe, Mar. 1997.

[89] L. Kalampoukas, A. Varma, and K. K. Ramakrishnan. An efficient rate allocation algorithm for ATM networks providing max-min fairness. In *HPN'95*, Palma de Mallorca, Spain, Sept. 1995. IFIP.

[90] L. Kalampoukas, A. Varma, and K. K. Ramakrishnan. Explicit window adaptation: A method to enhance TCP performance. In *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, San Fransisco, Mar. 1998.

[91] H. Kanakia, P. P. Mishra, and A. R. Reibman. An adaptive congestion control scheme for real time packet video transport. *IEEE/ACM Transactions on Networking*, 3(6), Dec. 1995.

[92] D. Katz. IP router alert option. Request for Comments (Proposed Standard) 2113, Internet Engineering Task Force, Feb. 1997.

[93] R. Kawamura and I. Tokizawa. Self-healing virtual path architecture in ATM networks. *IEEE Communications*, 33(9):72–78, Sept. 1995.

[94] S. Knowles. IESG advice from experience with path MTU discovery. Technical Report RFC 1435, Internet Engineering Task Force, Mar. 1993.

[95] I. Kouvelas, V. Hardman, and J. Crowcroft. Network adaptive continous-media applications through self organised transcoding. In *Proc. International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, Cambridge, England, July 1998.

[96] V. Kumar. *MBone: Interactive Multimedia On The Internet.* Macmillan Publishing (Simon & Schuster), 1995.

[97] K. Lai and M. Baker. Measuring bandwidth. In *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, New York, USA, Mar. 1999.

[98] T. V. Lakshman, A. Neidhardt, and T. J. Ott. The drop from front strategy in TCP and in TCP over ATM. In *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, San Fransisco, California, Mar. 1996.

[99] A. Y. Lao. Heterogeneous cell-relay network simulation and performance analysis with PTOLEMY. Technical Report UCB/ERL M94/8, Electronics Research Laboratory, UC Berkeley, 1993.

[100] D. C. Lee. Effects of leaky bucket parameters on the average queueing delay: Worst case analysis. In *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, Toronto, Canada, June 1994.

[101] K.-W. Lee, S. Ha, and V. Bharghavan. Irma: A reliable multicast architecture for the internet. In *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, New York, USA, Mar. 1999.

[102] A. Legout and E. W. Biersack. Fast convergence for cumulative layered multicast transmission scheme. Technical report, Eurecom, Sophia-Antipolis, France, Oct. 1999. under submission.

[103] S. Li, Xue Paul and M. Ammar. Layered video multicast with retransmissions (LVMR): Evaluation of hierarchical rate control. In *Infocom'98*, San Fransisco, 1998.

[104] X. Li and M. H. Ammar. Bandwidth control for replicated-stream multicast video. In *HPDC Focus Workshop on Multimedia and Collaborative Environments (Fifth IEEE International Symposium on High Performance Distributed Computing)*, Syracuse, New York, Aug. 1996. IEEE Computer Societey.

[105] D. Lin and R. Morris. Dyanmics of random early detection. In *Sigcomm'97*, october 1997.

[106] W. Luo and M. El Zarki. Transmitting scalable MPEG-2 video over based networks. In *First Workshop on ATM Management*, Paris, December 1995.

[107] J. Mahdavi and S. Floyd. TCP-friendly unicast rate-based flow control, June 1997. Technical note, available from http://ftp.ee.lbl.gov/floyd/papers.html.

[108] G. Malkin. RIP version 2 carrying additional information. Technical Report RFC 1388, Internet Engineering Task Force, Jan. 1993.

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

[110] M. Mathis, J. Semke, J. Mahdavi, and T. Ott. The macroscopic behavior of the TCP congestion avoidance algorithm. *IEEE Network*, 11(6), November/December 1997.

[111] S. McCanne. *Scalable Compression and Transmission of Internet Multicast Video.* PhD thesis, University of California Berkeley, Aug. 1996.

[112] S. McCanne and V. Jacobson. vic: A flexible framework for packet video. In *Proc. of ACM Multimedia '95*, Nov. 1995.

[113] S. McCanne, V. Jacobson, and M. Vetterli. Receiver-driven layered multicast. In *SIGCOMM Symposium on Communications Architectures and Protocols*, Palo Alto, California, Aug. 1996.

[114] S. McCanne, M. Vetterli, and V. Jacobson. Low-complexity video coding for receiver-driven layered multicast. *IEEE Journal on Selected Areas in Communications*, 16(6), Aug. 1997.

[115] D. L. Mills. Network time protocol (version 3) specification, implementation. Technical Report RFC 1305, Internet Engineering Task Force, Mar. 1992.

[116] E. Miloslavsky and A. Zakhor. Constant PSNR rate control for layered video compressing using matching pursuits. In *Proceedings of the International Conference on Image Processing*, Kobe, Japan, Oct. 1999.

[117] P. V. Mockapetris. Domain names - concepts and facilities. Technical Report RFC 1034, Internet Engineering Task Force, Nov. 1987.

[118] S. B. Moon, P. Skelly, and D. Towsley. Estimation and removal of clock skew from network delay measurements. In *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, New York, Mar. 1999.

[119] J. Moy. OSPF version 2. Technical Report RFC 1247, Internet Engineering Task Force, July 1991.

[120] J. Moy. MOSPF: analysis and experience. Technical Report RFC 1585, Internet Engineering Task Force, Mar. 1994.

[121] J. Nagle. Congestion control in IP/TCP internetworks. *ACM Computer Communication Review*, 14(4):11–17, Oct. 1984.

[122] B. Nandy, N. Seddigh, and P. Pieda. Diffserv's assured forwarding PHB: what assurance does the customer have? In *Proc. International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, Basking Ridge, New Jersey, June 1999.

[123] L. Nederlof, K. Struyve, C. O'Shea, M. H., Y. Du, and B. Tamayo. End-to-end survivable broadband networks. *IEEE Communications*, 33(9):63–69, Sept. 1995.

[124] J. Nonnenmacher and E. W. Biersack. Optimal multicast feedback. In *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, San Francisco, USA, Mar. 1998.

[125] A. Odlyzko. The economics of the internet: Utility, utilization, pricing and quality of service. Juli 1998.

[126] A. Olsen. Adaptive feedback compensation for distributed load-based routing systems in datagram packet-switched communications networks. *Computer Communication Review*, 27(3):83–99, July 1997.

[127] R. O. Onvural. *Asynchronous Transfer Mode Networks: Performance Issues.* Artech House, Boston, 1994.

[128] T. Ott, J. Kemperman, and M. Mathis. Window size behavior in TCP/IP with constant loss probability. June 1997.

[129] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP throughput: A simple model and its empirical validation. In *ACM SIGCOMM '98*, Vancouver, Oct 1998.

[130] J. Padhye, J. Kurose, D. Towsley, and R. Koodli. A TCP-friendly rate adjustment protocol for continuous media flows over best effort networks. Technical report, University of Massachusetts, Amherst, Massachusetts, Oct. 1998.

[131] P. P. Pan and H. Schulzrinne. YESSIR: A simple reservation mechanism for the Internet. In *Proc. International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, Cambridge, England, July 1998. also IBM Research Technical Report TC20967.

[132] A. K. Parekh and R. G. Gallager. A generalized processor sharing approach to flow control in integrated services networks – the multiple node case. In *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, volume 2, pages 521–530 (5a.1), San Francisco, California, March/April 1993. IEEE.

[133] K. Park, G. Kim, and M. Corvella. On the relationship between file sizes, transport protocols and self-similar network traffic. In *International Conference on Network Protocols (ICNP)*, Columbus, Ohio, Oct 1996.

[134] Parkinson, W. Bradford, and J. Spilker. *Global Positioning System: Theory and Practice.* American Institute of Aeronautics and Astronautics, Inc, Washington, DC, 1996.

[135] S. Paul, K. Sabnani, J. C. Lin, and S. Bhattacharyya. Reliable Multicast Transport Protocol (RMTP). *IEEE Journal on Selected Areas in Communications*, April 1997.

[136] V. Paxon. Empirically-derived analytic models of wide-area TCP connections. *IEEE/ACM Transactions on Networking*, 2(4), Augus 1994.

[137] V. Paxon. *Measurements and Analysis of End-to-End Internet Dynamics.* PhD thesis, Lawrence Berkeley National Laboratory, University of California, Berkeley, California, Apr. 1997.

[138] V. Paxson and S. Floyd. Wide-area traffic: The failure of paisson modeling. In *Sigcomm'95*. ACM Sigcomm, Juli 1995.

[139] V. Paxson and S. Floyd. Why we don't how to simulate the Internet. In *Proceeding of the 1997 Winter Simulation Conference*, December 1997.

[140] T. Pfeifer, T. Magedanz, and R. Popescu-Zeletin. Intelligent handling of communication media. In *Sixth IEEE Computer Society Workshop on Future Trends of Distributed Computing Systems (FTDCS)*, Tunis (Tunisia), Oct. 1997.

[141] J. Postel. User datagram protocol. Technical Report RFC 768, Internet Engineering Task Force, Aug. 1980.

[142] J. Postel. Internet protocol. Technical Report RFC 791, Internet Engineering Task Force, Sept. 1981.

[143] J. Postel. Transmission control protocol. Technical Report RFC 793, Internet Engineering Task Force, Sept. 1981.

[144] J. Postel and J. K. Reynolds. File transfer protocol. Technical Report RFC 959, Internet Engineering Task Force, Oct. 1985.

[145] K. K. Ramakrishnan and R. Jain. A binary feedback scheme for congestion avoidance in computer networks with a connectionless network layer. In *SIGCOMM Symposium on Communications Architectures and Protocols*, pages 303–313, Stanford, California, Aug. 1988. ACM.

[146] R. Rejaie, M. Handley, and D. Estrin. An end-to-end rate-based congestion control mechanism for realtime streams in the Internet. In *Infocom'99*, New York, March 1999. IEEE.

[147] I. Rhee, N. Balaguru, and G. N. Rouskas. MTCP: Scalable TCP-like congestion control for reliable multicast. In *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, New York, USA, Mar. 1999.

[148] L. Roberts, B. Makrucki, T. Tofigh, A. Barnhart, B. Holden, J. Diagle, M. Hulchyj, H. Suzuki, G. Ramamurthy, P. Newman, N. Giroux, R. Kositpaiboon, S. Sathe, G. Garg, and N. Yin. Closed-loop rate-based traffic management. Technical Report 94-0438R1, ATM Forum, July 1994.

[149] A. Romanow and S. Floyd. The dynamics of TCP traffic over ATM networks. In *SIGCOMM Symposium on Communications Architectures and Protocols*, pages 79–88, London, UK, Sept. 1994.

[150] D. Rubenstein, J. Kurose, and D. Towsley. The impact of multicast layering on network fairness. In *Special Interest Group on Data Communication (SIGCOMM'99)*, Cambridge, USA, Aug. 1999.

[151] H. Sanneck. Adaptive loss concealment for internet telephony applications. In *Proceedings INET '98*, Geneva, Switzerland, July 1998.

[152] H. Sanneck and G. Carle. Predictive loss pattern queue management for internet routers. In *Internet Routing and Quality of Service, S. Civanlar, P. Doolan, J. Luciani, R. Onvural, Editors, Proceedings SPIE Vol.3529A*, Boston, MA, November 1998.

[153] T. Sano, N. Yamanouchi, T. Shiroshita, and O. Takahashi. Flow and congestion control for bulk reliable multicast protocols. In *International Symposium on Voice, Video, and Data Communications (SPIE)*, Dallas, USA, Oct. 1998.

[154] K. Savetz, N. Randall, and Y. Lepage. *MBONE: Multicasting tomorrow's Internet.* IDG Books, Foster City, CA, 1995.

[155] H. Schulzrinne. A comprehensive multimedia control architecture for the Internet. In *Proc. International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, St. Louis, Missouri, May 1997.

[156] H. Schulzrinne. Internet problems and potential. In *Proc. of Global Information Infrastructure Workshop at IEEE International Conference on Communications (ICC)*, Montreal, Canada, June 1997.

[157] H. Schulzrinne. Re-engineering the telephone system. In *Proc. of IEEE Singapore International Conference on Networks (SICON)*, Singapore, Apr. 1997.

[158] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: a transport protocol for real-time applications. Technical Report RFC 1889, Internet Engineering Task Force, Jan. 1996.

[159] H. Schulzrinne and J. Rosenberg. Signaling for internet telephony. In *ICNP'98*, Austin, Texas, Oct. 1998.

[160] S. Shenker, C. Partridge, and R. Guerin. Specification of guaranteed quality of service. Technical Report RFC 2212, Internet Engineering Task Force, Sept. 1997.

[161] S. Shenker, L. Zhang, and D. D. Clark. Some observations on the dynamics of a congestion control algorithm. *ACM Computer Communication Review*, pages 30–39, Oct. 1990.

[162] S. S. Shirish. ATM Forum traffic management specification version 4.0. Technical Report 94-0013R6, ATM Forum, June 1995.

[163] D. Sisalem. Rate based congestion control and its effects on TCP over ATM. Diplomarbeit, School of Engineering, TU Berlin, Berlin, June 1995.

[164] D. Sisalem. Behavior of various switch mechanisms for the ABR service in the presence of persistent and dynamic traffic. In *The Fourth IEEE Workshop on the Architecture and Implementation of High Performance Communication Systems (HPCS'97)*, Chalkidiki, Greece, June 1997.

[165] D. Sisalem. End-to-end quality of service control using adaptive applications. In *IFIP Fifth International Workshop on Quality of Service (IWQOS '97)*, New York, May 1997.

[166] D. Sisalem. Fairness of adaptive multimedia applications. In *International Conference on Communications (ICC'98)*, Atlanta, USA, June 1998.

[167] D. Sisalem and F. Emanuel. QoS control using adaptive layered data transmission. In *IEEE Multimedia Systems Conference'98*, Austin, TX, USA, June 1998.

[168] D. Sisalem, S. Krishnamurthy, and S. Dao. DiffRes: A light weight reservation protocol for the differentiated services environment. Technical report, HRL Laboratories, Malibu, USA, Dec. 1999. under submission.

[169] D. Sisalem and H. Schulzrinne. End-to-end rate control in ABR. In *First Workshop on ATM Traffic Management (WATM'95)*, pages 281–287, Paris, France, Dec. 1995. IFIP WG 6.2.

[170] D. Sisalem and H. Schulzrinne. Binary congestion notification in TCP. In *Conference Record of the International Conference on Communications (ICC)*, Dallas, Texas, June 1996. IEEE.

[171] D. Sisalem and H. Schulzrinne. Congestion control in TCP: Performance of binary congestion notification enhanced TCP compared to Reno and Tahoe TCP. In *International Conference on Network Protocols (ICNP)*, pages 268–275, Columbus, Ohio, Oct. 1996.

[172] D. Sisalem and H. Schulzrinne. Switch mechanisms for the ABR service: A comparison study. In *Telecommunication Distribution Parallelism (TDP'96)*, La Londes Les Maures, France, June 1996. Institut National des Telecommunications, Evry.

[173] D. Sisalem and H. Schulzrinne. The loss-delay based adjustment algorithm: A TCP-friendly adaptation scheme. In *Proc. International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, Cambridge, England, July 1998.

[174] D. Sisalem and H. Schulzrinne. The multimedia internet terminal. *Journal of Telecommunication Systems*, 9(3):423–444, Sept. 1998.

[175] D. Sisalem and H. Schulzrinne. The adaptive load service: An ABR-like service for the Internet. In *Fifth IEEE Symposium on Computers and Communications (ISCC'2000)*, Juan Les Pins, France, July 2000.

[176] D. Sisalem, H. Schulzrinne, and C. Sieckmeyer. The network video terminal. In *HPDC Focus Workshop on Multimedia and Collaborative Environments (Fifth IEEE International Symposium on High Performance Distributed Computing)*, Syracuse, New York, Aug. 1996. IEEE Computer Society.

[177] D. Sisalem and A. Wolisz. Towards TCP-friendly adaptive multimedia applications based on RTP. In *Fourth IEEE Symposium on Computers and Communications (ISCC'99)*, pages 166–172, Egypt, July 1999. IEEE.

[178] D. Sisalem and A. Wolisz. MLDA: A TCP-friendly congestion control framework for heterogeneous multicast environments. In *Eighth International Workshop on Quality of Service (IWQoS 2000)*, pages 65–74, Pittsburgh, PA, June 2000. IEEE.

[179] W. R. Stevens. *TCP/IP illustrated: the protocols*, volume 1. Addison-Wesley, Reading, Massachusetts, 1994.

[180] B. Suter, T. Lakshman, D. Stiliadis, and A. Choudhury. Design considerations for supporting TCP with per-flow queueing. In *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, San Francisco, California, Mar. 1998.

[181] A. K. Talukdar, B. R. Badrinath, and A. Acharya. Rate adaptation schemes in networks with mobile hosts. In *ACM/IEEE MobiCom'98*, Dallas, Oct 1998.

[182] W. Tan and A. Zakhor. Multicast transmission of scalable video using receiver-driven hierarchical FEC. In *Packet Video Workshop 99*, New York, Apr. 1999.

[183] D. Taubman and A. Zakhor. Multi-rate 3-D subband coding of video. *IEEE Transactions on Image Processing*, 3(5):572–588, Sept. 1994.

[184] D. Tennenhouse, J. M. Smith, D. Sincoskie, D. J. Wetherall, and J.Gary. A survey of active network research. *IEEE Commumnications Magazine*, 35:80–86, Jan. 1997.

[185] K. Thompson, G. J. Miller, and R. Wilder. Wide-area Internet traffic patterns and characteristics. *IEEE Network*, 11(6):–, November/December 1997.

[186] M. E. Thyfault. Resurgence of convergence. *Information Week*, page 50ff, Apr. 1998.

[187] T. Turletti, S. F. Prisis, and J.-C. Bolot. Experiments with a layered transmission scheme over the Internet. Rapport de recherche 3296, INRIA, Nov. 1997.

[188] C. J. Van den Branden Lambrecht and O. Verscheure. Perceptual quality measure using a spatio-temporal model of the human virsual system. In *SPIE'96*, volume 2668, pages 450–461, San Jose, CA, February 1996.

[189] O. Verscheure, P. Frossard, and M. Hamdi. MPEG-2 video services over packet networks: Joint effect of encoding and data loss on user-oriented QoS. In *Proc. International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, Cambridge, England, July 1998.

[190] M. Vetterli and K. Uz. Multiresolution coding techniques for digital video: a review. *Special Issue on Multidimensional Processing of Video Signals, Multidimensional Systems and Signal Processing*, (3):161–187, 1992.

[191] L. Vicisano, L. Rizzo, and J. Crowcroft. TCP-like congestion control for layered multicast data transfer. In *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, San Francisco, USA, Mar. 1998.

[192] D. Waitzman, C. Partridge, and S. E. Deering. Distance vector multicast routing protocol. Technical Report RFC 1075, Internet Engineering Task Force, Nov. 1988.

[193] G. K. Wallace. The JPEG still picture compression standard. *Communications ACM*, 34(4):30–44, Apr. 1991.

[194] H. A. Wang and M. Schwartz. Achieving bounded fairness for multicast and tcp traffic in the internet. In *SIGCOMM'98*, vancouver, Canada, September 1998.

[195] Z. Wang and J. Crowcroft. A new congestion control scheme: Slow start and search (tri-s). *ACM Computer Communication Review*, 21(1):32–43, Jan. 1991.

[196] Z. Wang and J. Crowcroft. A dual-window model for flow and congestion control. *The Distributed Computing Engineering Journal, Institute of Physics/British Computer Society/IEE*, 1(3):162–172, Sept. 1994.

[197] A. Watson and M. A. Sasse. Measuring perceived quality of speech and video in multimedia conferencing applications. In *ACM Multimedia '98*, 1998.

[198] C. T. Webster, Arthur A. Jones and M. H. Pinson. An objective video quality assessment system based on human perception. In *In SPIE Human Vision, Visual Proceeding and Digital Display IV*, volume 1913, pages 15–26, San Jose, CA, February 1993.

[199] J. Wroclawski. Specification of the controlled-load network element service. Technical Report RFC 2211, Internet Engineering Task Force, Sept. 1997.

[200] L. Wu, R. Sharma, and B. Smith. Thin streams: An architecture for multicating layered video. In *Proc. International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, St. Louis, USA, May 1997.

[201] C.-Q. Yang and A. V. S. Reddy. A taxonomy for congestion control algorithms in packet switching networks. *IEEE Network*, 9(4), July/August 1995.

[202] N. Yeadon. *Quality of Service Filtering for Multimedia Communications*. PhD thesis, Lancaster University, Lancaster, UK, May 1996.

[203] L. Zhang, S. Shenker, and D. D. Clark. Observations on the dynamics of a congestion control algorithm: The effects of two-way trffic. In *SIGCOMM Symposium on Communications Architectures and Protocols*, pages 133–147, Z rich, Switzerland, Sept. 1991. ACM. also in *Computer Communication Review* 21 (4), Sep. 1991.