End-to-End Quality of Service Over Heterogeneous Networks

Torsten Braun • Michel Diaz •
José Enríquez-Gabeiras • Thomas Staub

End-to-End Quality of Service Over Heterogeneous Networks

Springer

Prof. Dr. Torsten Braun
Thomas Staub
Universität Bern
Institut für Informatik und
Angewandte Mathematik (IAM)
Neubrückstr. 10
3012 Bern
Switzerland
braun@iam.unibe.ch
staub@iam.unibe.ch

Michel Diaz
LAAS - CNRS
7 Avenue du Colonel Roche
31400 Toulouse
France
diaz@laas.fr

José Enríquez Gabeiras
Telefónica I+D
Emilio Vargas 6
28043 Madrid
Spain
jeg@tid.es

ISBN 978-3-540-79119-5 e-ISBN 978-3-540-79120-1

DOI 10.1007/978-3-540-79120-1

Library of Congress Control Number: 2008926817

ACM Computing Classification (1998): C.2.2

Cover design: KünkelLopka GmbH, Heidelberg

Printed on acid-free paper

9 8 7 6 5 4 3 2 1

springer.com

Preface

The Internet has evolved from an academic network for data applications such as file transfer and net news, to a global general-purpose network used for a variety of different applications—electronic mail, voice over IP, television, peer-to-peer file sharing, video streaming and many more. The heterogeneity of applications results in rather different application requirements in terms of bandwidth, delay, loss, etc. Ideally, the underlying network supports Quality-of-Service parameters such that applications can request the desired services from the network and do not need to take actions by themselves to achieve the desired communication quality. Initially, the Internet was not designed to support Quality of Service, and only in the last decade have appropriate mechanisms been developed. Those mechanisms operate mainly on the Internet Protocol (IP) level, but also network-specific mechanisms—e.g., targeted to particular wired/wireless access network technologies—are required.

The goal of the European 6th Framework Programme (FP6) Integrated Project "End-to-end Quality of Service Support over Heterogeneous Networks" (EuQoS) was to develop, implement and evaluate concepts and mechanisms to support QoS end-to-end, meaning that QoS mechanisms in end systems, access networks, interdomain links and within domains must be supported. The EuQoS project developed an impressive set of innovative solutions and novel scientific ideas to support end-to-end QoS on the Internet. New mechanisms and concepts were designed and implemented in a European-wide distributed testbed. In addition to the rather technical design and implementation work, the project also developed training material introducing basic QoS mechanisms and techniques. Several e-learning modules were developed and are currently being used at several partner universities for teaching on MSc or PhD levels.

The significant technical and educational results achieved during the EuQoS project motivated us to use the gained knowledge and experiences of the project partners and write this book on end-to-end QoS in heterogeneous IP networks. The book basically consists of three parts. In Chaps. 1–4, we discuss QoS mechanisms and protocols such as scheduling schemes, QoS architectures, metrics and measurement techniques, traffic engineering and signalling protocols, and the latest standardisation activities. In Chap. 5, we describe related work and recent developments in the area of transport protocols, in particular how TCP can be optimised toward QoS support and fairness. The EuQoS system presented in Chap. 6 extends and combines the basic mechanisms discussed in the previous chapters. We show how a combination of different QoS-enabling mechanisms and protocols can be used and extended to build a comprehensive end-to-end QoS architecture over heterogeneous wired/wireless access networks. To evaluate QoS mechanisms and architectures, appropriate evaluation schemes are required. The two chapters in the appendix describe how simulation—in particular the well-known network simulator ns-2—as well as emulation techniques can be used for tests and evaluations.

This book, which is based on the achievements of the EuQoS project, would not have been possible without funding from the European Commission, as well as the tremendous efforts and enthusiasm of all the people involved in the project. Special thanks to Mark Günter for proofreading the text contributions to this book.

Torsten Braun
Michel Diaz
José Enríquez Gabeiras
Bern, Toulouse, Madrid. January 2008. *Thomas Staub*

Acknowledgements

The book editors and authors thank all people who were involved in the EuQoS project:

- **Telefonica I+D:** José Enríquez Gabeiras, María Ángeles Callejo Rodríguez, Francisco Javier Ramón Salguero, Gerardo García de Blas, Jorge Andrés Colás, Francisco Romero Bueno, Jesús Bravo Álvarez, María Luisa García Osma, Antonio J. Elizondo Armengol
- **University of Pisa (CPR/UoPisa):** Enzo Mingozzi, Giovanni Stea, Luciano Lenzini, Luca Bisti, Claudio Cicconetti, Linda Martorini, Abraham Gebrehiwot, Simone Bisogni, Paolo Sozzi, Alessandro Spanò, Luca Luigetti
- **Elsag Datamat:** Enrico Angori, Giuseppe Martufi, Marco Carusio, Alessandro Giorgini, Andrea Paselli, Giovanni Saccomandi, Marco Mauro
- **CNRS (LAAS-CNRS, ENSICA):** Michel Diaz, Florin Racaru, Ernesto Exposito, Philippe Owezarski, Patrick Sénac, Christophe Chassot, Nicolas Larrieu, Laurent Dairaine, Mathieu Gineste, Nicolas Van Wambeke, Slim Abdellatif, Sébastien Ardon, Roberto Willrich, Guillaume Auriol, Silvia Farraposo
- **France Telecom R&D:** Olivier Dugeon, Walid Htira, Michel Bourbao, Pascal Le Guern, Jean-Louis Le Roux, Stéphane Statiotis, Régis Fréchin, Claire Teisseire
- **Polska Telefonia Cyfrowa (ERA):** Michal Obuchowicz, Robert Parzydo, Adam Flizikowski, Edyta Rafalska, Karol Jez, Krzysztof Horszczaruk, Krzysztof Bronarski, Krzysztof Samp, Maciej Rozowicz, Michal Dudzinski, Pawel Caban, Piotr Zadroga, Slawomir Tkacz
- **Martel:** Martin Potts, Mark Guenter, Sandra Wittwer
- **NICTA:** Emmanuel Lochin, Guillaume Jourjon, Sebastien Ardon, Ernesto Exposito, Feiselia Tan, Laurent Dairaine
- **PointerCom:** Roberto Marega, Stefano Salsano, Donald Papalilo, Gianluca Martiniello, Valeria Calcagni

- **Polish Telecom R&D:** Zbigniew Kopertowski, Jaroslaw Kowalczyk, Tomasz Ciszkowski
- **Portugal Telecom Inovação (PTIN):** Jorge Carapinha, Nuno Carapeto, Paulo Loureiro, Arnaldo Santos, Eduardo Silva, Fernando Santiago, Helena Paula Matos, Hugo Manaia, Isabel Borges, Jacinto Barbeira, Filipe Peixinho.
- **Red Zinc:** Donal Morris, Brian Widger, Diarmuid O Neill, Léa Compin, Oscar Cuidad
- **Silogic:** Laurent Baresse, Benoît Baurens, Jean-Philippe Darmet, Yannick Lizzi, François Meaude
- **INDRA (previously SOLUZIONA):** Ignacio Fresno, Jaime Orozco, Luis Collantes Abril, Pablo Vaquero Barbón, Rubén Romero San Martín, Jorge Alonso, Maria Lurdes Sousa, Raul Manzano Barroso
- **Telscom AG:** Sathya Rao, Marcin Michalak
- **Technical University of Catalonia (UPC):** Jordi Domingo-Pascual, Loránd Jakab, Marcelo Yannuzzi, René Serral-Gracià, Xavier Masip-Bruin
- **University of Bern:** Torsten Braun, Thomas Staub, Dragan Milić, Marc Brogle, Marc-Alain Steinemann, Thomas Bernoulli, Gerald Wagenknecht, Markus Wulff, Patrick Lauer, Markus Anwander, Matthias Scheidegger
- **University of Paderborn/C-LAB:** Isabell Jahnich, Achim Rettberg, Chris Loeser, Michael Ditze, Kay Klobedanz, Sebastian Seitz, Andreas König, Volker Spaarmann, Matthias Grawinkel
- **University of Rome:** Antonio Pietrabissa, Francesco Delli Priscoli, Sabrina Giampaoletti, Emiliano Guainella, Erasmo Di Santo, Gianfranco Santoro, Ilaria Marchetti, Massimiliano Rossi
- **Universidade de Coimbra:** Edmundo Monteiro, Luís Cordeiro, Bruno Carvalho, Fernando Boavida, Gabriela Batista Leão, Isidro Caramelo, Jian Zhang, Jorge Sá Silva, Marilia Curado, Maxwel Carmo, Paulo Simões, Romulo Ribeiro, Vitor Bernardo, David Palma, Rui Vilão, Luís Conceição
- **Warsaw University of Technology:** Wojciech Burakowski, Andrzej Beben, Halina Tarasiuk, Jaroslaw Sliwinski, Jordi Mongay Batalla, Marek Dabrowski, Piotr Krawiec, Robert Janowski
- **Ericsson:** Antoine de Poorter, Julio López Roldan, Miguel Angel Recio, Jesus Renero Quintero, José Luis Agundez
- **Hospital Divino Espirito Santo:** José Manuel Ponte, António Vasco Viveiros, Carlos P. Duarte, Paula Maciel, José M. Jesus Silva, Maura Medeiros, Maria Dulce Raposo

Contents

Acronyms

The following list contains acronyms used in the book and their explanation. Most acronyms can be found in the index as well together with a page reference.

ALM	Application layer multicast
API	Application programming interface
CIDR	Classless Internet domain routing
COPS	Common Open Policy Service
DVMRP	Distance-vector multicast routing protocol
IGMP	Internet group management protocol
IP	Internet protocol
IPTV	Internet Protocol Television
IPv4	Internet protocol version 4
IPv6	Internet protocol version 6
ISP	Internet service provider
MM	Multicast Middleware
MOSPF	Multicast open shortest path first
NSIS	Next Step In Signalling
P2P	Peer-to-peer
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PIM	Protocol-independant multicast
QoS	Quality of Service
SDP	Session Description Protocol
SE	Signalling Entities
SIP	Session initiation Protocol
SSQ	Synchronize State Query
TCP	Transmission control protocol

TTL	Time-To-Live
UAC	User Agent Client
UAS	User Agent Server
UDP	User datagram protocol
VLSM	Variable length subnet mask
MPLS	Multi Protocol Label Switching
TE	Traffic Engineering
RIP	Routing Information Protocol
IGP	Interior Gateway Protocol
OSPF	Open Shortest Path First
IS-IS	Intermediate System-Intermediate System
RSVP	ReSerVation Protocol
IETF	Internet Engineering Task Force
FEC	Forwarding Equivalence Class
LSR	Label-Switching Router
LFIB	Label Forwarding Information Base
ATM	Asynchronous Transfer Mode
DLCI	Data-Link Connection Identifier
VPI	Virtual Path Identifier
VCI	Virtual Channel Identifier
BGP	Border Gateway Protocol
LDP	Label Distribution Protocol
LIB	Label Information Base
LSP	Label-Switched Path
CBR	Constraint-Based Routing
TED	Traffic Engineering Database
CSPF	Constrained Shortest Path First
SPF	Shortest Path First
CR-LDP	Constraint-based Routing Label Distribution Protocol
TSPEC	Traffic Specification
ERO	Explicit Route Object
BA	Behavior Aggregate
PHB	Per Hop Behavior
DSCP	Diff-Serv Codepoint
OA	Ordered Aggregate
PSC	PHB Scheduling Class
AF	Assured Forwarding
E-LSP	EXP-Inferred-PSC LSP
L-LSP	Label-Only-Inferred-PSC LSP
BE	Best Effort
DS-TE	Diff-Serv-aware Traffic Engineering
CT	Class Type
BC	Bandwidth Constraint
MAM	Maximum Allocation Bandwidth Constraints Model
RDM	Russian Doll Bandwidth Constraints Model

AC	Access Category
ADSL	Asymmetric DSL
AIFS	Arbitrary Inter-Frame Space
AP	Access Point
AS	Autonomous Systems
ASPB	AS Path Builder
ATM	Asynchronous Transfer Mode
BR	Border Router
BRAS	Broadband Remote Access Server
BRPC	Backward Recursive Path Computation
CAC	Connection Admission Control
CBR	Constant Bit Rate
CoS	Class of Service
CPE	Customer Premises Equipment
CRA	Continuous Rate Assignment
CW	Contention Window
DAMA	Demand Assignment Multiple Access
DCF	Distributed Coordination Function
DSL	Digital Subscriber Line
DSLAM	Digital Subscriber Line Access Multiplexer
DVB-S	Digital Video Broadcasting - Satellite
DVB-RCS	Digital Video Broadcasting - Reverse Channel Satellite
e2e CoS	End-to-end Class of Service
EDCA	Enhanced Distributed Coordination Access
ER	Edge Router
ES	Ethernet Switch
FCA	Free Capacity Assignment
FTP	File Transfer Protocol
GGSN	GPRS Gateway Support Node
HTD	High Throughput Data
IPLR	IP Packet Loss Ratio
IPTD	IP Packet Transfer Delay
IPDV	IP Packet Delay Variation
MAC	Medium Access Control
MT	Mobile Terminal
NCC	Network Control Centre
NRT	Non Real Time
OGGSN	Open GPRS Gateway Support Node
PCC	Path Computation Client
PCE	Path Computation Element
PCEP	PCE Protocol
PQ	Priority Queuing
PR	Peak Rate
RA	Resource Allocator
RBDC	Rate Based Dynamic Capacity

RM	Resource Manager
RNC	Radio Network Controller
RT	Real Time
SHDSL	Symmetrical High Bitrate DSL
SLA	Service Level Agreement
ST	Satellite Terminal
STD	Standard
TERO	Traffic Engineering and Resource Optimization
TOS	Type of Service
UTRAN	UMTS Terrestrial Radio Access Network
VBDC	Volume Based Dynamic Capacity
VBR	Variable Bit Rate
VDSL	Very High Bitrate DSL
VoD	Video on Demand
VoIP	Voice over IP
VTC	Video Teleconference
WFQ	Weighted Fair Queueing
WMM	WiFi Multi-Media
WRED	Weighted Random Early Detection
WRR	Weighted Round-Robin
CLI	Command Line Interface
EQ-BGP	Enhanced QoS Border Gateway Protocol
QoS NLRI	QoS Network Layer Reachability Information
DoP	Degree of Preference
SNMP	Simple Network Management Protocol
TMN	Telecommunications Management Network
SAAA	Security, Authentication, Authorization and Accounting
QoSR	Quality of Service Routing
xDSL	Digital Subscriber Line
UMTS	Universal Mobile Telecommunications System
LAN	Local Area Network
NREN	National REsearch Network
GEANT	Multi-gigabit pan-European data communications network
NTI	Network Technology Independent
NTD	Network Technology Dependent
AQ-SSN	Application Quality Signalling and Service Negotiation
CHAR	CHARging module QCM Quality Control Module
MMS	Monitoring and Measurement System
EQ-SAP	EQ-Service Access Point
PQ-WFQ	Priority Queueing - Weighted Fair Queueing
SCTP	Stream Control Transmission Protocol
DCCP	Datagram Congestion Control Protocol
ETP	Enhanced Transport Protocol
gTFRC	TCP-Friendly Rate Congestion Control
TC	Time Constraints

SACK	Selective ACKnowledgement
PCMA	Pulse Code Modulation a-law
PCMU	Pulse Code Modulation mu-law
CIF	Common Intermediate Format
QCIF	Quarter Common Intermediate Format
SQCIF	Sub Quarter Common Intermediate Format
e2e	end-to-end

1

Motivation and Basics

Torsten Braun and Thomas Staub

Summary. This chapter provides an introduction to the topic of Quality of Service and motivates the rest of the chapters in the book. The performance of network applications (e.g., video on demand, collaboration tools, voice over IP, Internet TV, video conferencing) depends on the quality of network connections. Parameters such as packet loss, delay, delay variation, and out-of-order delivery are important to describe network performance. Since applications differ in their Quality-of-Service (QoS) needs, this chapter provides a classification of some typical applications and sheds light on their requirements. It further illustrates the implementation and performance of QoS-aware applications, as well as the benefits of such QoS-aware applications, and concludes with a short overview of the following chapters.

1.1 Quality of Service and its Parameters

Quality of Service (QoS) is a measure of the ability of network and computing systems to provide different levels of services to selected applications and associated network flows. Since Internet Protocol (IP) based networks are expected to form the basis for all kinds of future communication services such as data transfer, telephony, television, etc., and users expect at least the same quality for those services such as when delivered over dedicated networks, QoS support for IP networks is urgently required. Currently, however, IP networks are Best-Effort networks. As the name suggests, packet forwarding is performed with the best effort, but without guaranteeing bandwidth, delay bounds etc.

Before going into more detail, the term QoS must be defined more accurately. [1] distinguishes perceptual, application, system, network and device QoS [2] for multimedia systems. The network QoS parameters are most important for this book. They may be specified in terms of the network load including packet interarrival times, burstiness and packet sizes, as well as the network performance describing network service guarantees. Network performance can be described in more detail by several parameters such as delay, delay variation, bandwidth, and packet loss rate. These are the basic QoS parameters discussed hereafter. QoS supporting systems try to guarantee QoS by not exceeding QoS parameter limits. QoS parameters are also called QoS metrics in relation to measurement, see Chap. 2.

1.1.1 Delay and Delay Variations in End-to-End Packet Delivery

As described more formally in Chap. 2, the one-way delay (only called delay in this chapter) normally describes the average delay that packets experience over a specific connection. Packet delays can be split into four components:

- *Processing delay* is the time needed by network elements such as routers or end systems to process a packet. It depends on the processing speed of the network element hardware and the complexity of the functions to perform. These range from simple packet classification for forwarding and fire-walling, to complex payload modifications for encryption and content adaptation.
- Network components normally have input and/or output queues. The time a packet resides in these queues is called *queuing delay*. Queues become larger when the network becomes congested, which results in a longer queuing delay.
- *Transmission delay* is the time needed to transmit a packet at a specific bit rate. It can be calculated as

$$\text{transmission delay} = \frac{\text{number of bits to transmit}}{\text{transmission rate}}.$$

- The *propagation delay* describes the time needed by the signals to travel (propagate) through the medium. It can be calculated as

$$\text{propagation delay} = \frac{\text{physical distance}}{\text{propagation velocity}}.$$

Propagation and queuing delay are the key contributors to delay as long as no heavy processing like encryption or packetisation by applications (cf. Sect. 1.2.2.1) is needed.

In real-world networks, packets experience a delay on their path from the sender to the receiver, which is not constant but rather varying over time, because conditions on a route and the involved systems change. This is a result of the fluctuation of Internet traffic and resulting queue sizes. The delay is bounded by a minimum and maximum delay. The difference between these bounds is called delay variation. In the remainder of the section we use the same definition for the delay variation as the ITU-T for the IPDV as outlined in Chap. 2. The typical behaviour for the delay of packets of a single flow in the Internet is depicted in Fig. 1.1. Since processing, transmission, and propagation delay normally do not change for a given route, the delay variation has its source in the varying queuing delay.

The delay variation can be compensated by buffering packets, either within the network elements (routers) or the receiving end systems. Since end-system memory is much cheaper than router memory, buffering in the end system is usually preferred. Figure 1.2 shows the concept of play-out buffering. In this example, we assume that a continuous stream of packets is sent with a difference of 160 ms between each packet. Each packet has a timestamp indicating its transmission time. The delay of

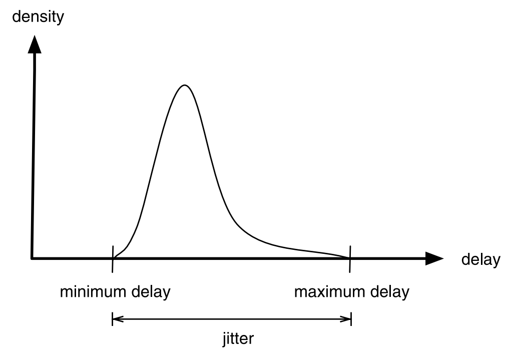

Fig. 1.1. Delay variation

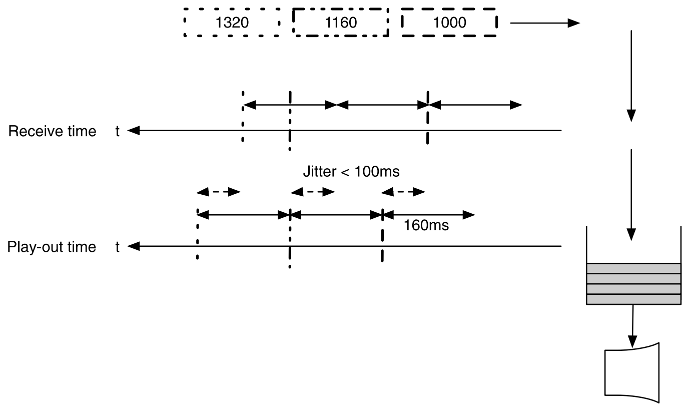

Fig. 1.2. Play-out buffer

the first and third packet is rather low (minimum delay), while the second packet experienced maximum delay. Assuming that the difference between the minimum and the maximum delay (delay variation) is not more than 100 ms, it is sufficient to delay each packet by 100 ms at the receiver. In this case, the first packet is delayed by additional 100 ms after reception and the second one arrives just in time so that the two packets can be played out with the original difference of 160 ms. The example shows that play-out buffering only works if the delay variation is bounded.

1.1.2 Bandwidth and Packet Loss Ratio

The bandwidth describes the capacity of a link or end-to-end path. It is measured in bits per second. The packet loss rate indicates the number of packets that do not reach the destination in relation to all sent packets. Packet loss has mainly two causes—packet errors, e.g. due to bad link quality (especially on wireless links) and packet drops, e.g. due to congestion. It can be calculated as

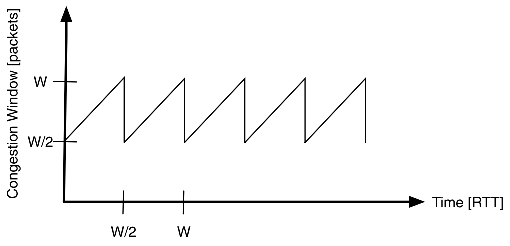

Fig. 1.3. Congestion window in TCP

$$\text{Packet loss ratio} = \frac{\text{packets sent} - \text{packets received}}{\text{packets sent}}.$$

Considering IP packets, there is no direct relation between bandwidth and packet loss ratio. However, in case of TCP connections, the achievable bandwidth depends on the round-trip time (RTT) and the packet error rate: $BW < \frac{MSS}{RTT} \times \frac{1}{\sqrt{p}}$ [3, 4] where BW is bandwidth; MSS is maximum segment size; RTT is round-trip time; and p is packet error rate. The achievable bandwidth can be calculated as follows. It is assumed that the delivery of $\frac{1}{p}$ packets is followed by a single packet loss, e.g. due to congestion. If the receiver acknowledges each packet, the window opens by 1 per round trip. With the maximum congestion window size W, and $\frac{W}{2}$ as the minimum congestion window in equilibrium, we get the behavior depicted in Fig. 1.3.

In each cycle ($= RTT \times \frac{W}{2}$) one packet is lost and the number of packets delivered is $(\frac{W}{2})^2 + \frac{1}{2}(\frac{W}{2})^2 = \frac{3}{8}W^2 = \frac{1}{p}$. The bandwidth BW is calculated as

$$\begin{aligned} BW &= \frac{\text{data per cycle}}{\text{time per cycle}} \\ &= \frac{MSS \times \frac{3}{8}W^2}{RTT \times \frac{W}{2}} \\ &= \frac{\frac{MSS}{p}}{RTT \times \sqrt{\frac{2}{3}p}} \\ &= \frac{MSS \times C}{RTT \times \sqrt{p}}, \quad C = \sqrt{1.5}. \end{aligned}$$

We conclude that the achievable bandwidth for a TCP connection depends on the round-trip time, as well as on the error rate.

1.2 Applications' QoS Requirements

Many network applications work fine with Best-Effort services, while others have strong QoS requirements and only work with guaranteed QoS, or at least benefit significantly if QoS guarantees are possible. After describing two classification schemes of network applications, we give an overview of application requirements for audio, video and data applications.

1.2.1 Types of Network Applications

Each application probably has individual QoS requirements. However, they can be classified using different classification schemes. In the following we discuss two classification schemes for network applications, namely (in)elastic applications as well as (non-)interactive applications.

1.2.1.1 Elastic and Inelastic Applications

First, we distinguish between elastic and inelastic applications. An elastic application is able to adapt to changing QoS parameters and does not fail in that case. Elastic applications are also called Best-Effort applications. File transfer and e-mail are examples of elastic applications, because they do not require guaranteed bandwidth or delay bounds. In case of low bandwidth, the file or e-mail transfer just takes somewhat longer.

In contrast to elastic applications, inelastic applications need strict QoS guarantees. Real-time applications by nature are mostly inelastic, but may have some ability to adapt to certain QoS parameter changes. For example, audio/video conferencing can adapt to less bandwidth by using more efficient video codecs. A codec (derived from "coder/decoder") is a software or hardware device able to encode and decode a digital data stream. However, a minimum bandwidth is required to run the most efficient codec. Moreover, the delay requirements are quite stringent in such cases.

1.2.1.2 Interactive and Noninteractive Applications

Another classification scheme distinguishes interactive and noninteractive applications. Interactive applications include human interaction. Typically, a human user interacts remotely with another end system and expects a quick reaction to the performed action. The reaction should normally be as quick as possible with hard bounds of low delay. Due to the strict delay bounds, the error rate is quite important, because retransmissions are not possible without exceeding the tolerable delay if round-trip times are rather high. Forward error correction is a means to reduce delays in such a case, but additional processing is required by the end systems. Figure 1.4 shows the delay and error rate requirements for real-time voice transmissions. Interactive applications can also be categorised as real-time applications. In most cases, they are inelastic too.

Examples of interactive applications are voice over IP (VoIP), audio/video conferencing, collaborative online applications, and online games. Examples of noninteractive applications are Web browsing, file transfer, chats and multimedia streaming. For multimedia streaming a server begins to send a continuous stream of multimedia data to be played out at the receiver. In theory, insufficient bandwidth and

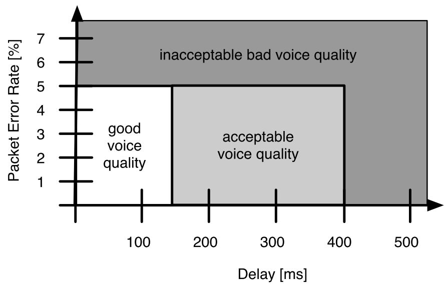

Fig. 1.4. Delay and error rate requirements for voice transmissions according to one-way values of [5]

delay variation can be compensated by buffering. If the bandwidth requirements are not met, the stream can only be played out after a significant delay introduced by buffering a huge amount of data, which can easily exceed available buffer space. As discussed in Sect. 1.1.1, large delay variation also has an impact on the required buffer size.

1.2.2 QoS Requirements of Applications

1.2.2.1 Audio Applications

Audio transmissions normally have widely varying bandwidth requirements, depending on whether telephony or high-fidelity music is being transferred. In addition to the encoded audio data, protocol overhead by IP, User Data Datagram (UDP), and Real-Time Transport Protocol (RTP) headers must be considered. Larger packets can reduce this overhead. In this case, several audio samples are collected and put together into a single packet, but longer packetisation intervals increase the delay. Moreover, sensitivity to packet loss is increased, since a single packet includes a rather long sequence of consecutive audio samples.

In particular, in case of interactive audio such as telephony, strong delay requirements exist. The recommended maximum tolerable delay for telephony is 150 ms [6]. This includes the four delay components described in Sect. 1.1.1 as well as packetisation delay. The delay variation should be limited too, since high delay variation increases the required size of play-out buffers at the receiver. Moreover, interactive audio applications are quite sensitive to packet loss. Thus, they require (very) low loss rates. The concrete numbers depend on the type of the audio application. Telephony using mother language requires less stringent error bounds than telephony using a foreign language or even high-fidelity music. Moreover, less efficient encodings have some degree of redundancy and can therefore tolerate higher packet loss.

In the case of streamed audio where a single user receives and listens to an audio stream without having interaction, the delay, bandwidth and error rate requirements can be relaxed. However, due to buffer limitations, guaranteeing QoS parameters might be desirable as well.

1.2.2.2 Video Applications

Many statements for audio apply to video transmissions as well. Audio and video transmissions have much in common. If interactive, both are sensitive to delay, delay variation and packet loss. However, there are several differences between audio and video. Most importantly, the required bandwidth for video is much higher, which depends highly on the quality level desired by the user or supported by the video equipment. While PC- or mobile hand-held-based video conferencing systems work with even a few tens of kbps, high-definition television demands several Mbps. The bandwidth requirements depend on several system parameters such as colour depth, screen size and resolution, frame rate and acceptable quality degradation by compression.

Another observation is that video traffic is usually burstier than audio traffic due to the used encoding schemes. Schemes like MPEG periodically send a so-called intracoded frame, which does not have any reference to other preceding or succeeding frames. Frames that are sent between these intracoded frames, so-called intercoded frames, can have a reference to other frames and only encode the difference (in terms of movement vectors, colour differences etc.) compared to the referred frames. This results in so-called weakly regular traffic for video, while audio often is strongly regular as depicted in Fig. 1.5. Weakly regular traffic creates some short-term bursts. The first option to handle such bursts is to provide sufficient resources (in particular bandwidth and buffer memory) in the network elements. The other option is to smooth out the traffic at the sender's end in order to produce rather constant traffic. However, this can again can lead to additional delays.

As for audio, the required video quality heavily depends on the type of the application scenario. A simple video conference with some known colleagues might have less stringent quality requirements than a movie or telemedicine applications. There is also a difference between stored and real-time video. If the video has been recorded in advance, video encoding can be more efficient resulting in higher burstiness, while live video compression may be less bursty due to the lack of processing time required for highly efficient interframe coding. The same as discussed for streamed audio applies to streamed video, but again, due to the higher bandwidth, buffer size requirements are much higher.

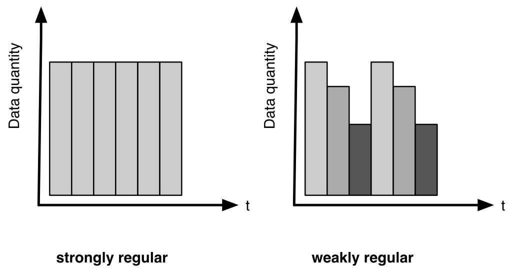

Fig. 1.5. Strongly and weakly regular traffic according to [1]

1.2.2.3 Data Traffic

Although nonaudio and nonvideo traffic is summarised as data traffic it must be kept in mind that there are many different data traffic classes. File transfers of multi-megabyte files as well as interactive console traffic is part of the data traffic category. The term transactional traffic is often used for interactive data traffic. Since there are thousands of data applications, it is impossible to discuss them all. Every application potentially has a unique traffic pattern and network requirements. Even different versions of the same application may result in very different traffic patterns [7].

1.3 Packet Scheduling in Network Elements

Sharing of network resources automatically introduces the problem of contention. Applications need QoS guarantees and congestion in networks is still common. Thus, scheduling disciplines implemented in network elements such as routers and switches are important so that network resources are shared fairly and performance guarantees for performance-critical applications. A scheduling discipline has to achieve two main tasks, as indicated by Fig. 1.6.

- First, it has to decide the order in which requests are serviced (packet selection).
- Second, it has to manage the service queue of requests awaiting service (packet dropping).

This section describes the concepts and requirements of a scheduling discipline and introduces some of the most important scheduling algorithms.

1.3.1 (Non)Work-Conserving Scheduling Disciplines

The most simple scheduling discipline is First Come First Serve (FCFS), also known as First In First Out. In this case, all arriving packets are served according to their arrival sequence. A scheduling discipline like FCFS is idle only if its queue is empty. Such schedulers are also called work-conserving [8]. On the other hand, a nonwork-conserving scheduling discipline may be idle even if it has packets to serve. A packet is sent only if it is eligible, otherwise it is delayed until it becomes eligible. However, why are nonwork-conserving scheduling disciplines useful? The reason is that downstream traffic can be made more predictable, and thus the buffer sizes and delay variation can be reduced in downstream routers and receiving systems. Bursts are eliminated and traffic becomes smoother. Of course, the mean queuing delay of a nonwork-conserving scheduling discipline is larger than with FCFS.

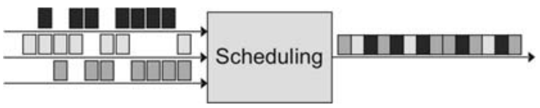

Fig. 1.6. Scheduling

The Conservation Law states that a work-conserving scheduling discipline can reduce a connection's mean delay, compared with FCFS, only at the expense of another connection. Thus, if a particular connection receives a lower delay than with FCFS, at least one other connection gets a higher delay. For that reason, the sum of delays with FCFS is a tight lower bound for the sum of delays for every scheduling discipline, whether it is work-conserving or not. The Conservation Law is as follows: $\sum \rho_i q_i = const$, with ρ_i as the mean utilisation of a link due to connection i and q_i as connection i's mean waiting time at the scheduler.

1.3.2 Fairness

Another important issue is fairness. A scheduling discipline should divide resources fairly among a set of users. A problem to be solved is when some users demand fewer resources than others do. So how can the resources left by these users be divided? The Max-Min Fair Share approach solves this problem by maximising the minimum share of a source whose demand is not fully satisfied. In this case, resources are allocated in the order of increasing demands. No user gets a resource share larger than its demand and sources with unsatisfied demand get an equal share of the resource. The resource allocation algorithm is as follows:

1. Divide capacity C by n: $\frac{C}{n}$ resources for each connection.
2. Connection 1 needs x_1 resources ($x_1 < \frac{C}{n}$).
3. Distribute exceeding resources $\frac{C}{n} - x_1$ equally among other connections, so that each connection gets $\frac{C}{n} + \frac{\frac{C}{n} - x_1}{(n-1)}$ resources allocated.
4. Continue the process if resource allocation is larger than x_2.

If we want to give a bigger share to some user than to others, we can use weights that reflect their relative resource share. The demands of the users are then normalised by the weight and sources with an unsatisfied demand get resource shares in proportion to their weights.

1.3.2.1 Requirements for Scheduling Disciplines

There are four (sometimes contradictory) requirements that a scheduling discipline must satisfy.

- Ease of implementation: In a high-speed network, once every few microseconds a server has to decide on which packet to pick for transmission. Therefore, a scheduling discipline should require only a few simple operations. Preferably, they should be implementable inexpensively in terms of hardware. Furthermore, the number of operations should be as independent of the number of scheduled connections as possible.
- Fairness and protection: An allocation at a network element can be considered as fair if it satisfies the max-min fair share criterion. For Best-Effort connections fairness is an intuitively desirable property. However, for guaranteed-service connections it is not a concern. A scheduling discipline provides protection if the

misbehaviour of one connection does not affect the performance of other connections.

- Performance bounds: A scheduling discipline should allow arbitrary connection performance bounds, limited by the conservation law only. Performance bounds can be defined in a deterministic or statistical way.
- Ease of efficiency and admission control: A scheduling discipline should be able to decide whether it is possible to meet the performance bounds of new connections or not. This decision is also called admission control. The decision should lead neither to network underutilisation, nor to jeopardising the performance of existing connections.

1.3.3 Scheduling Disciplines

In addition to FCFS, a variety of scheduling disciplines exist. An ideal and work-conserving scheduling discipline that provides a max-min fair allocation is Generalised Processor Sharing (GPS). Unfortunately, GPS cannot be implemented. GPS serves packets as if they are in separate logical queues. Each nonempty queue is visited in turn and an infinitesimally small amount of data in each queue is served. Thus, the scheduler can visit each queue at least once in any finite time interval. We assume N connections with equal weights send data to the scheduler infinitely fast. The GPS scheduler serves an infinitesimally small amount from each connection in turn. Therefore, each connection gets a share of $\frac{1}{N}$ of the bandwidth. If a connection sends less data than this share, the queue of this scheduler will occasionally be empty. Thus, the GPS scheduler skips empty queues, and because of its round-robin service the time saved is equally distributed to the other connections resulting in a new service rate. If another connection has a rate larger than $\frac{1}{N}$, but smaller than the new service rate, its queue will occasionally be empty too. Again, the remaining connections will receive a slightly larger share. Obviously, each connection with a rate smaller than its fair share gets its demand allocated, while each connection with a larger demand gets an equal share. Thus, we see that GPS service achieves the max-min fair share defined above.

A priority scheduler knows different priority levels, which have their own queue. Every incoming packet will be assigned to a priority level, depending on protocol type, application, IP addresses, etc. The packet with the highest priority will be processed. Packets with lower priority are selected only if there are no packets with higher priority available.

A (Weighted) Round-Robin scheduler has a queue for every service class and serves the packets from each nonempty buffer in turn. To obtain a service differentiation, service classes can have different weights so that the buffers will be served in proportion to their weight.

Weighted Fair Queuing is an emulation of GPS scheduling. For each incoming packet a finish number is calculated. The theoretical meaning of this finish number is the time the last bit of the packet should be transmitted if the GPS scheduler would be used. In practice, the finish number is only a service tag and does not stand for the actual time at which a packet is served. Packets are ordered by their finish number and serviced in that order. The finish number of a packet arriving at an inactive connection is the sum of the current round number and the packet size in bits. If the packet is arriving at an active connection, the finish number is the sum of the largest finish number in the queue, or of the packet last served and the packet size in bits. The finish number is calculated as [8]:

$$F(i,k,t) = \max\left\{F(i,k-1,t),\, R(t)\right\} + P(i,k,t)/\phi(i), \quad \text{with}$$

$F(i,k,t)$: finish number of the kth packet of connection i with arriving time t
$P(i,k,t)$: packet length
$R(t)$: number of the round at time t
$\phi(i)$: weight of connection i.

The round number increases inversely proportional to the number of active connections. It indicates the number of rounds a bit-by-bit round robin scheduler has completed at a given time. A connection is active if the largest finish number in the connections queue or of the packet last served is bigger than the current round number.

Rate-controlled scheduling consists of two components: a regulator and a scheduler. The regulator determines the packets eligibility time and forwards only eligible packets to the scheduler. The scheduler uses an arbitrary algorithm (FIFO, Priority, Round Robin etc.) to schedule the packets.

1.3.4 Packet Dropping

The limited length of the various queues in a scheduler requires dropping packets in overload conditions. In order to avoid that important packets are dropped while less-important packets are not dropped, packets can be marked by applications or routers with a packet-drop priority. Packets with high drop priorities are dropped first. One approach could be to assign a low dropping priority to packets that have been travelling for a very long time. This avoids dropping packets that have already consumed a large amount of network resources.

Another issue is whether a scheduler drops packets when there is absolutely no space in the queue, or somewhat earlier to always have some space for important packets to serve. Alternatives are early dropping schemes such as Random Early Detection (RED). In this case, packets can be dropped even if the queue is not full. This always keeps some space for important packets arriving later. Those packets would otherwise have to be dropped.

When packets must be dropped, we can drop them from the tail of the queue. This is easy to implement, but may be unfair if packets of well-behaving connections have just arrived. An alternative is random dropping, where packets to be dropped are selected randomly. Even packets at the head of the queue can be dropped. This scheme has the advantage that the receiver will notify a packet loss earlier than for dropping packets at the tail. In this case, the congestion control mechanisms as implemented in TCP will react earlier.