Adaptive Delay-constrained Internet Media Transport

Dissertation

zur Erlangung des Grades des
Doktors der Ingenieurwissenschaften (Dr.-Ing.)
der Naturwissenschaftlich-Technischen Fakultäten
der Universität des Saarlandes

vorgelegt von

Manuel Gorius

Saarbrücken, September 2012

Tag des Kolloquiums

19. Dezember 2012

Dekan der Naturwissenschaftlich-Technischen Fakultät I

Prof. Dr. Mark Groves
Universität des Saarlandes, Saarbrücken, Deutschland

Prüfungsausschuss

Prof. Dr. Jan Reineke (Vorsitzender des Prüfungsausschusses)
Universität des Saarlandes, Saarbrücken, Deutschland
Prof. Dr.-Ing. Thorsten Herfet
Universität des Saarlandes, Saarbrücken, Deutschland
Prof. Dr.-Ing. Ralf Steinmetz
Technische Universität Darmstadt, Darmstadt, Deutschland
Prof. Dr.-Ing. Ulrich Reimers
Technische Universität Braunschweig, Braunschweig, Deutschland
Dr.-Ing. Tim Schwartz
Universität des Saarlandes, Saarbrücken, Deutschland

Berichterstatter

Prof. Dr.-Ing. Thorsten Herfet
Universität des Saarlandes, Saarbrücken, Deutschland
Prof. Dr.-Ing. Ralf Steinmetz
Technische Universität Darmstadt, Darmstadt, Deutschland
Prof. Dr.-Ing. Ulrich Reimers
Technische Universität Braunschweig, Braunschweig, Deutschland

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus anderen Quellen oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet. Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder ähnlicher Form in einem Verfahren zur Erlangung eines akademischen Grades vorgelegt.

Manuel Gorius

Short Abstract

Reliable transport layer Internet protocols do not satisfy the requirements of packetized, real-time multimedia streams. The available thesis motivates and defines *predictable reliability* as a novel, capacity-approaching transport paradigm, supporting an application-specific level of reliability under a strict delay constraint. This paradigm is being implemented into a new protocol design – the *Predictably Reliable Real-time Transport* protocol (PRRT). In order to predictably achieve the desired level of reliability, proactive and reactive error control must be optimized under the application's delay constraint. Hence, predictably reliable error control relies on stochastic modeling of the protocol response to the modeled packet loss behavior of the network path. The result of the joined modeling is periodically evaluated by a *reliability control policy* that validates the protocol configuration under the application constraints and under consideration of the available network bandwidth. The adaptation of the protocol parameters is formulated into a *combinatorial optimization problem* that is solved by a fast search algorithm incorporating explicit knowledge about the search space. Experimental evaluation of PRRT in real Internet scenarios demonstrates that predictably reliable transport meets the strict QoS constraints of high-quality, audio-visual streaming applications.

Kurze Zusammenfassung

Zuverlässige Internet-Protokolle auf Transport-Layer erfüllen nicht die Anforderungen paketierter Echtzeit-Multimediaströme. Die vorliegende Arbeit motiviert und definiert *Predictable Reliability* als ein neuartiges, kapazitäterreichendes Transport-Paradigma, das einen anwendungsspezifischen Grad an Zuverlässigkeit unter strikter Zeitbegrenzung unterstützt. Dieses Paradigma wird in ein neues Protokoll-Design implementiert – das *Predictably Reliable Real-time Transport* Protokoll (PRRT). Um prädizierbar einen gewünschten Grad an Zuverlässigkeit zu erreichen, müssen proaktive und reaktive Maßnahmen zum Fehlerschutz unter der Zeitbegrenzung der Anwendung optimiert werden. Daher beruht Fehlerschutz mit Predictable Reliability auf der stochastischen Modellierung des Protokoll-Verhaltens unter modelliertem Paketverlust-Verhalten des Netzwerkpfades. Das Ergebnis der kombinierten Modellierung wird periodisch durch eine *Reliability Control* Strategie ausgewertet, die die Konfiguration des Protokolls unter den Begrenzungen der Anwendung und unter Berücksichtigung der verfügbaren Netzwerkbandbreite validiert. Die Adaption der Protokoll-Parameter wird durch ein *kombinatorisches Optimierungsproblem* formuliert, welches von einem schnellen Suchalgorithmus gelöst wird, der explizites Wissen über den Suchraum einbezieht. Experimentelle Auswertung von PRRT in realen Internet-Szenarien demonstriert, dass Transport mit Predictable Reliability die strikten Auflagen hochqualitativer, audiovisueller Streaming-Anwendungen erfüllt.

Abstract

Reliable transport layer Internet protocols do not satisfy the requirements of packetized, real-time multimedia streams. First, major limitations result from their primary design objective of serving total reliability without tolerating residual packet loss. This property leads to unpredictable delivery delay on lossy network paths and conflicts with the strict rendering deadlines of multimedia services that explicitly prefer timeliness over reliability. Second, the strict layering of the ISO/OSI network stack prevents applications to communicate their specific quality of service (QoS) requirements to the transport layer. Consequently, transport protocols do not provide an interface for the negotiation of constraints on packet loss and delivery delay. Third, as the provision of scalable one-to-many transport requires careful design – especially under combination with error control – it is insufficiently supported by reliable protocols. Yet broadcast or multicast distribution of digital media is efficient and not unusual. As of today these issues are clearly unsolved in the prevalently HTTP/TCP-based media streaming such that the available Internet bandwidth is significantly underutilized and the presentation quality suffers severely.

The available thesis motivates and defines *predictable reliability* as a novel, capacity-approaching transport paradigm, supporting an application-specific level of reliability under a strict delay constraint. This paradigm is being implemented into a new protocol design – the *Predictably Reliable Real-time Transport* protocol (PRRT). The protocol combines the fundamental concepts of proactive and reactive packet-level error control into an adaptive hybrid error coding architecture. The flexibility of the hybrid scheme enables the protocol to adaptively follow the dynamic capacity of the packet-erasure channels generated by a wide range of Internet protocol infrastructures. Combined with packet loss notifications via negative acknowledgments, it provides capacity-approaching coding efficiency in point-to-point as well as one-to-many transmission scenarios.

In order to predictably achieve the desired level of reliability, proactive and reactive error control must be optimized under the application's delay constraint. Hence, predictably reliable error control relies on stochastic modeling of the protocol response to the network path's packet loss behavior. A *block-erasure model* captures the characteristics of the packet loss process. Further, a *protocol performance model* is being developed that predicts the protocol's residual packet loss rate as well as its coding overhead based on the statistical representation of the network state. The performance model reflects the efficiency of one-to-many error control and incorporates the impact of unreliable delivery of the negative acknowledgments. The result of the joined modeling is periodically evaluated by a *reliability control policy* that validates the protocol configuration under the application constraints and under consideration of the available network bandwidth. The adaptation of the protocol parameters is formulated into a *combinatorial optimization problem* that is solved by a fast search algorithm incorporating explicit knowledge about the search space.

Experimental evaluation of PRRT in real Internet scenarios demonstrates that predictably reliable transport meets the strict QoS constraints of high-quality, audio-visual streaming applications. In particular, broadcast services over Internet Protocol require packet streams to be delivered at a residual loss rate of 10^{-6} to 10^{-5} under a delay constraint of few hundred milliseconds, depending on their degree of interactivity. Within different experiments, the protocol implementation is evaluated at the transport of high-quality broadcast TV via Internet Protocol. Especially wired wide area network paths

as well as wireless and mobile networks expose the transport protocol to highly dynamic packet loss rates and propagation delays. Comparative experiments with recent advancements in *dynamic Internet video streaming* confirm PRRT's significant gain in efficiency. The fairness towards existing transport protocols on shared network paths is demonstrated under delay-based congestion control.

Acknowledgment

During the proceeding of this thesis I had the pleasure to interact and cooperate with an endless list of great and valuable people, which turned quiet but also demanding times during the past few years into enjoyable moments.

First of all I would like to thank Prof. Dr.-Ing. Thorsten Herfet for providing me the opportunity to accomplish this research work under his supervision at the Telecommunications Lab of Saarland University. Without doubt, this work has been significantly shaped due to his in-depth comments and the numerous inspiring discussions we had. Further, I would like to express my special thanks to both Prof. Dr.-Ing. Ralf Steinmetz and Prof. Dr.-Ing. Ulrich Reimers for contributing their reviews on this thesis.

Warmest thanks go to all present and former members of the Telecommunications Lab at Saarland University for creating a pleasant and inspiring working environment throughout the past years and for being the best colleagues I could imagine. In particular, I would like to greatly thank Goran Petrovic, Jochen Miroll, Michael Karl and Yongtao Shuai for co-authoring results of joint research work into several publications. I am very grateful to Jochen Grün for the numerous discussions on the protocol design developed within this thesis and his invaluable contribution to the open source implementation. In addition, special thanks are due to Jochen Miroll, Eric Haschke and Michael Karl for the exciting cooperation during numerous project implementations. This thesis would not have evolved as far if it was not for the fruitful scientific discussions with Goran Petrovic, Jochen Miroll, Jochen Krämer, Christopher Haccius and Zhao Li. Similarly, I want to express a great acknowledgment to Zakaria Keshta and Diane Chlupka for being always available in case of technical and organizational questions. At this position I should not forget to mention the members of the "TC Lab Band", Anne Dehof, Christopher Kaiser, Thorsten Herfet, Christopher Haccius and Frank Waßmuth, for making all those Tuesday evenings a special and balancing event within the day-to-day working routine.

It was definitely among the most enlightening moments of the past few years to be able to work together with many dedicated and talented students on their Bachelor's and Master's theses. Therefore, I would like to acknowledge Michael Karl, Jochen Grün, Robert Gogolok, Martin Emrich, Kurnia Hendrawan, Bernd Wittefeld, Bo Fu, Yongtao Shuai, Juhi Kulshrestha and Hanjo Viets with warmest thanks. It was a real pleasure to cooperate with them while exploring new fields and ideas in the research related to this thesis and it was always a driving factor for its progress and development.

I would like to thank all those people not being listed so far that directly and indirectly contributed to my research work. These also include the anonymous reviewers whose comments significantly helped to improve the presentation of the topic and the experimental results. Beyond that, parts of the underlying research work of this thesis have been carried out under the ITC3D project funded by the Intel Visual Computing Institute Saarbruecken.

Finally, I would like to express my greatest appreciation and gratitude to my family, all above my parents, Eva and Franz-Josef Gorius, for their endless and steady support during demanding times and for having enabled and significantly prepared this kind of carreer to me.

Contents

List of Figures

List of Tables

List of Algorithms

Nomenclature

Abbreviations

3GPP	3rd Generation Partnership Project
AIMD	Additive Increase/Multiplicative Decrease
AP	Access Point
ARQ	Automatic Repeat reQuest
AVPF	RTP Audio-Visual Profile with Feedback
CDN	Content Delivery Network
CoS	Class of Service
CRC	Cyclic Redundancy Check
DASH	Dynamic Adaptive Streaming over HTTP
DCCP	Datagram Congestion Control Protocol
DPLL	Digital Phase-Locked Loop
DVB-H	Digital Video Broadcasting – Handhelds
ECN	Explicit Congestion Notification
EWMA	Exponentially-Weighted Moving Average
FEC	Forward Error Coding
FTT	Forward Trip Time
GCC	Group Correlation Coefficient
GE	Gilbert-Elliott (block-erasure model)
GF	Galois Field
GPLR	Group Packet Loss Rate
GPS	Global Positioning System
GRTT	Group Round Trip Time
H-ARQ	Hybrid Automatic Repeat reQuest
HEC	Hybrid Error Coding

HSDPA	High Speed Downlink Packet Access
HTTP	Hypertext Transfer Protocol
IETF	Internet Engineering Task Force
i. i. d.	independently and identically distributed
IMS	IP Multimedia Subsystem
IP	Internet Protocol
IPTV	Internet Protocol Television
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
ISO	International Organization for Standardization
ISP	Internet Service Provider
LAN	Local Area Network
LTE	Long Term Evolution
MAC	Medium Access Control
MBMS	Multimedia Broadcast Multicast Service
MDS	Maximum Distance Separable
MLE	Maximum-Likelihood Estimation
NACK	Negative Acknowledgment
NCO	Numerically Controlled Oscillator
NORM	NACK-oriented Reliable Multicast
NTP	Network Time Protocol
NUM	Network Utility Maximization
OSI	Open Systems Interconnection Reference Model
P2P	Peer-to-Peer
PLR	Packet Loss Rate
PR-DCCP	Datagram Congestion Control Protocol with Partial Reliability
PRRT	Predictably Reliable Real-time Transport
PR-SCTP	Stream Control Transmission Protocol with Partial Reliability
PTP	Precision Time Protocol

QoS	Quality of Service
RTCP	Real-Time Control Protocol
RTP	Real-Time Transport Protocol
RTT	Round Trip Time
SACK	Selective Acknowledgment
SCTP	Stream Control Transmission Protocol
SIP	Session Initiation Protocol
SLA	Service Level Agreement
SNR	Signal-to-Noise Ratio
TCP	Transmission Control Protocol
TV	Television
UDP	User Datagram Protocol
VoD	Video-on-Demand

Symbols

a	accuracy
A_c	random variable for the aggregate block length at the receiver after repair cycle c
α, β, γ	filter coefficients
A_{ufb}	random variable for the aggregate block length sent after repair cycle c under unreliable feedback
B	bad network state
c	repair cycle index with $0 \leq c \leq N_C$
C	channel capacity
C_{lt}	loss-tolerant channel capacity
$\vec{c}$	code word (vector)
D_C	erasure coding delay
D_{DTx}	source packet transmission delay
δ	deviation
D_{FB}	feedback timer
D_{FEC}	forward error coding delay

$D_{FEC,min}$	minimum forward error coding delay
d	feedback repetition factor
D_{PL}	packet loss detection delay
D_{PR}	propagation delay
D_{PTx}	repair packet transmission delay
D_Q	queueing delay
D_{REQ}	request timeout
D_{RS}	response delay
D_{SUP}	maximum feedback suppression timer
D_T	application delay constraint
D_{TO}	packet timeout
e	erasure
ϵ	reordering threshold
F_c	random variable for the number of receivers requiring to send a loss notification in cycle c
F_r	random variable determining whether a loss notification is sent
$f_T(T)$	timer distribution for feedback suppression
G	good network state
G_{EC}	generator matrix of the erasure code
ι	input symbol
I	input alphabet
k	coding block length (number of source packets per coding block)
λ	arrival rate
L_b	transmission burst length
L_D	source packet length
L_{DHdr}	length of source packet header
L_g	erasure gap length
L_{LFB}	length of loss notifications
L_{NFB}	length of network state feedback packet

L_P	length of parity packet
L_{PHdr}	length of repair packet header
M	queue size
m	sequence length
μ	service rate
$\vec{m}$	message (vector)
N	sequence length
n	code word length
$n[c]$	aggregate code word length after cycle c with $n = n[N_C]$
N_C	number of repair cycles
$N_{C,max}$	maximum number of repair cycles
N_{FB}	number of redundant loss notifications
N_P	schedule of repair packets
$n_{P,max}$	maximum number of repair packets per cycle
N_R	number of receivers
ν	defines the confidence interval as a multiple of the standard error
Ω	output alphabet
P_B	stady state probability of bad network state
P_{BB}	self-transition probability of bad network state
P_C	confidence level
$P_{cycle}(c)$	probability of reaching repair cycle c
$P_d(n, k, i, j)$	probability of hitting j source packets if i erasures affect a sequence of n coded packets including k source packets
P_e	packet erasure probability
P_f	feedback loss probability
$P_{fail}(c)$	probability of decoding failure after repair cycle c
P_G	steady state probability of good network state
P_{GG}	self-transition probability of good network state
ϕ	percentile of the receiver group covered by predictable reliability

P_j	steady-state probability of queue level j
$P_m(e, m)$	block error distribution, probability of e erasures in a sequence of length m
$P_{mfb}(A_c, N_C)$	probability mass function of the aggregate block length under N_C repair cycles
P_r	residual packet erasure probability
$P_{r,min}$	minimum residual packet erasure rate
$P_{r,ufb}$	residual packet erasure rate under unreliable feedback
$Pr(X)$	probability of event X
P_T	application reliability constraint
$P_{mfb}(n[c], P_f, N_C)$	probability mass function of the aggregate block length under feedback loss rate P_f and N_C repair cycles
P_v	virtual erasure probability
q	alphabet size
R	code rate
R_C	network capacity
R_d	dynamic code rate
R_{eff}	effective rate of the transport protocol (protocol throughput)
ρ	correlation coefficient (of the Gilbert-Elliott model)
RI	redundancy information
RI_{min}	minimum redundancy information
R_{LFB}	rate of loss notifications
R_{max}	maximum code rate
R_{NFB}	rate of network state feedback
R_{PO}	rate of the protocol overhead
R_S	media source rate
R_T	rate constraint
S	sequence (local definition)
s	sequence number or local index
τ	step size in the estimation of the optimum code word length

t_{FB}	time of sending feedback (loss notification)
θ	system utilization (queueing model)
T_{NFB}	interval of network feedback
t_r	time of receiving source packet
T_r, T_s	feedback suppression timer at receiver r, s
T_S	source packet interval
t_s	time of sending source packet
t_w	time of writing source packet into network socket
X	local random variable
x, i, j, b	local variables
$\zeta_{c,d}$	increment of the repair packet schedule from cycle c to cycle d

Functions

$\bar{x}$	average of x
$\hat{x}$	estimate of x
$E(X)$	expected value of random variable X
$H(X)$	entropy of random variable X
$H(X \mid Y)$	conditional entropy of random variable X, given Y
$H(z)$	low pass filter
$I(X;Y)$	mutual information of random variables X and Y
$\max_{x \in X} x$	maximum of x
$\min_{x \in X} x$	minimum of x
$\neg x$	negation of x
$\sigma(X)$	variance of random variable X
$\vec{x}$	vector x

1. Introduction

"To err is human, to be error-tolerant is divine."
Stephen M. Casner

Communication between an information source and a remote receiver has always been subject to physical barriers. Over centuries the distance between the communicating endpoints used to be the limiting constraint. This limitation has been virtually eliminated by the invention of wired and wireless telephony and data transmission systems, since those are approaching signal propagation at the speed of light. The Internet has meanwhile evolved to the primary network of digital communication. Yet being built upon various methods of physical signal transmission, it provides an abstraction layer that enables a virtually unlimited number of applications in digital communications between virtually arbitrary locations.

Shape and content of the global network have changed significantly throughout the past decade. The network is no longer an instance of pure host-to-host communication. The presence of powerful search engines and the availability of huge amounts of user-generated content are about to turn it into a content-centric network [80, 34]. Applications of the so-called Web 2.0 render the Internet a medium of daily communication through voice and video telephony services, entertainment services and social networking. The popularity of such high-rate, low-latency audiovisual services is ever growing along with the user's demand for excellent quality. Meanwhile, the packet-based transmission of multimedia content causes more than half of the traffic in the World Wide Web. According to Cisco's Visual Networking Index, more than 90% of the global consumer traffic will be caused by video communications such as Internet Protocol Television (IPTV), Video-on-Demand (VoD), Internet Television (Internet TV), and peer-to-peer video by 2014[1]. In near future, the lack of scalable, efficient transport and error control is believed to become the main innovation bottleneck for such services.

Common to all these applications is a requirement for high-speed and low-latency transport between the data acquisition and rendering endpoints. At the same time, the loss of few transmission units is tolerable due to their transient nature. As such services will be the predominant content in the future Internet, efficient transport protocols are required that adapt to their specific requirements in order to preserve a high-quality multimedia experience at the consumer's end device. Furthermore, this should be achieved under optimum utilization of the available link capacity since provisioning of Internet bandwidth as a service will continue to be expensive.

This thesis motivates and defines *predictable reliability* as a smart transport concept for delay-constrained and loss-tolerant communications services over Internet Protocol (IP). The following chapter introduces into the topic and formulates the corresponding problem statement. A survey of related work in the field of reliable and partially reliable

[1] http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360_ns827_Networking_Solutions_White_Paper.html

data transport is given. Starting from an abstract definition of predictable reliability as a newly introduced transport paradigm, an overview is provided over the components of the protocol architecture developed within this thesis. This architecture is fundamental for the remainder of the thesis since protocol design as well as analysis in later chapters are based on it and each focus on one essential component of the overall architecture of predictable reliability.

1.1. Internet Media Transport

Despite the fact that the Internet Protocol has not been designed to deliver real-time media services at high data rate, it is meanwhile widely deployed as an alternative carrier infrastructure for such content. As a result of the increasing popularity of IP-based communication, a shift in paradigm towards *everything over IP* [84] becomes evident. The Internet Protocol becomes increasingly important in the digital media broadcast where the delivery of live media content via IP infrastructure is already widely deployed.

The requirements of the media transport differ significantly from those of file transfer. In particular, a continuous packet stream is offered to the network that needs synchronous handling at the end points despite the asynchronous nature of the IP infrastructure. Guidelines on individual requirements of interactive and non-interactive audio and video communications services have been standardized under the ITU-T Y.1541 quality of service (QoS) classes [137] with mainly the following dimensions:

- **Packet Loss Rate:** Compressed audiovisual media have a specific tolerance for packet loss. In case of video transmission the reliability requirements are strict. For instance, the broadcast-compliant transport of IPTV requires visual artifacts to be limited to maximum one per hour, which corresponds to a packet loss rate of 10^{-6} [53, 155]. Audio streams have smaller transmission units and sophisticated codecs can compensate for lost transmission units via interpolation such that packet loss up to a small percentage is tolerable depending on the application.

- **Transport Delay:** The transport delay constraints differ between interactive and non-interactive multimedia applications. While interactivity is considered to be degraded as soon as the bidirectional delay exceeds 150 to 250 milliseconds, non-interactive services might tolerate a few seconds [137]. Especially for conversational applications the end-to-end delay and the delay variation should be bounded since the end devices implement limited buffer space.

- **Bandwidth:** Multimedia streams have rigid bandwidth requirements. Depending on their source coding algorithm, the media source rate is constant or variable over time [153]. A continuous rendering of the media stream is not possible if the bandwidth requirement is not fulfilled.

1.1.1. Problem Statement

The fundamental objective of the IP media transport can be stated as finding a network path with sufficiently large bandwidth and low latency between arbitrary endpoints and efficiently utilizing the bandwidth available on this path. Further, the transport protocol should not place restrictions on the location or the connecting infrastructure while delivering the media service. These requirements conflict with several aspects of the current

Internet architecture. In particular, this thesis addresses shortcomings of transport layer protocols for efficient and continuous media delivery. Open issues as well as undesired properties of available transport protocols are discussed in the following.

Bounded Delivery Time and Predictable Packet Loss Rate

Packet-switching Internet is a *best-effort service* implementing reliability on transport layer. As a result of the transport layer error control, the delivery delay of each single packet is hardly predictable. Audiovisual applications prefer timeliness before reliability such that they can tolerate a small amount of residual errors. They require the transport protocol to trade reliability for timeliness. *Partial reliability* extensions have been designed for available transport protocols in order to enable bounded delivery delay for loss-tolerant payload. However, they do not allow to explicitly control the residual packet loss rate at the same time.

Available transport protocols do not provide an interface to receive specific QoS constraints from the application. The required QoS is only provided on *managed Internet* paths, where over-provisioning in resource reservation ensures that the transport protocol meets the application constraints. Outside of managed networks, continuous and timely delivery of media streams cannot be guaranteed.

Heterogeneity and Transparency

With the broad availability of broadband wireless and mobile networks, Internet paths become increasingly heterogeneous. Since IP makes the Internet path's heterogeneity transparent to the upper layers, the transport protocol is subject to highly variable *network state*[2]. Reliable transport protocols suffer from significant loss in throughput on wireless networks, which introduce physical packet corruption in addition to packet loss by queue overflow. As a result, the QoS requirements of multimedia streams cannot be satisfied continuously.

Given the dynamics and heterogeneity of today's Internet paths, it is a fundamental question of this thesis, whether audiovisual and interactive services can be delivered with adequate QoS over unmanaged Internet. Considering the amount of user-generated multimedia content that is shared beyond the borders of the service provider's network infrastructure, it is obvious that such content cannot benefit from managed Internet. Consequently, it is the responsibility of the transport protocol to mediate between the variable network state and the constant application requirements as good as possible. The transport protocol should render the network state transparent to the application such as the Internet protocol abstracts the underlying infrastructure.

Efficient Error Control

The expected scarceness of bandwidth and spectrum has motivated the research in channel coding over several decades to develop capacity-approaching coding schemes. Despite the fact that transport layer reliability is channel coding in information-theoretic sense, efficiency has received less attention during the design of transport layer protocols in the past. In general, reliable protocols introduce a considerable amount of header and

[2]This thesis refers to the joint observation of round trip delay and packet loss characteristics as the network state.

signaling overhead and do not adapt the error control to the temporally variable network state.

As a result of strict layering in the Internet stack, available protocols operate in an agnostic fashion, i. e. they ignore valuable information about the application requirements and the network state. Meanwhile, equation-based protocol design is a strong trend in networking research that tends to observe the IP network stack holistically [31]. Equation-based design formulates error control and bandwidth allocation as a distributed optimization problem, where each layer contributes a partial solution. In fact, the Internet's bidirectional communication is the key to adaptive channel coding, which allows the source to refine transport parameters according to the receiver's observation of the network state in an adaptive error control function. Suchlike schemes require accurate modeling of the network state as well as performance modeling of the transport protocol in order to simulate the protocol's behavior.

Congestion Control and Protocol Fairness

As audiovisual communications services become the dominant traffic source in the future Internet, their stable coexistence and their fairness towards background traffic is an active research topic. The Internet implements flow multiplexing in a decentralized fashion. In order to provide fair access to a wide range of protocols and applications, congestion control algorithms have been implemented into reliable transport protocols. However, reliable protocols – most of all the Transmission Control Protocol (TCP) – have been designed to carry *elastic* data traffic that does not have strict delivery time constraints. This property allows the protocol to determine a dynamic window size per round trip time (RTT) in order to limit the throughput for flow and congestion control.

In contrast, multimedia traffic is *inelastic* by nature and renders error control as well as congestion control techniques difficult. Rate reduction implies the loss of transmission units or reduced quality for media flows. The behavior of such flows under congestion control is not yet well understood. Common result of the research conducted in this field is the observation that high-rate media streams require the goodput to be stable over longer periods [23, 11, 58], which results in reduced responsiveness to congestion events. Especially today's dynamic streaming applications based on TCP require overprovisioning of the available bandwidth by up to 100 % of the media source rate in order to achieve smooth rendering at the receiving device [158, 71].

Scalable Distribution

Despite the increasing individuality in media consumption, a considerable amount of multimedia content is still received in a broadcast-like fashion, i. e. simultaneously by large groups of customers. *IP Multicast* implements the idea of one-to-many delivery with constant complexity at the source in that the network infrastructure scales the signal with the number of active participants. In contrast to classical media broadcast it can cover a subset of networked receivers via Internet group management. However, point-to-point reliability cannot easily be extended to whole multicast groups since especially the receiver feedback to the sender scales linearly with the group size.

Under Internet Protocol version 4 (IPv4) [124], multicast has not experienced its breakthrough in wide area networking as it is optionally supported in this standard. Internet

Protocol version 6 (IPv6) [48], however, defines multicast as a mandatory protocol feature. Nevertheless, as a result of the scalability issues in error control and the insufficient multicast support in the available Internet infrastructure, the option for multicast is missing in most of today's reliable transport protocols. Scalable multimedia distribution is therefore mostly implemented on application layer by establishment of content delivery networks (CDN) and peer-to-peer (P2P) communication [77, 98, 14, 12]. Yet both approaches have significant disadvantages compared to IP multicast. Whereas the former requires hardware and network resources to scale linearly with the number of customers, the latter suffers from peer churn, discontinuous service and low throughput. Whenever large-scale media distribution is addressed in wireless and mobile networks, where the medium is shared and bandwidth is scarce, multicast solutions are desired and therefore part of mobile broadcast standards such as Digital Video Broadcasting – Handhelds (DVB-H) [52] and Multimedia Broadcast Multicast Service (MBMS) [1].

1.1.2. Contribution

A comprehensive, media-friendly transport layer protocol architecture is being developed throughout this thesis. The architecture includes contributions in the fields of

- *capacity-approaching packet-level erasure coding*
- *predictably reliable protocol design*
- *stochastic protocol performance modeling*
- *reliability control and congestion control.*

The protocol is specifically designed to address the above problem statement within the components introduced in the following.

Packet-Level Error Control

In order to provide flexible error control, a *dynamic, packet-level block-erasure coding* scheme is being designed and implemented. The scheme is based on packet-level hybrid error coding (HEC), which has been found to be an efficient approach for loss-tolerant erasure coding on bidirectional packet erasure channels [132, 147]. Forward error coding (FEC) in terms of optimal erasure codes [129] and the reactive transmission of repair packets closely interact under those codes, both contributing their individual advantages. In particular, the HEC enables capacity-approaching error control under dynamic network state.

Since error control significantly influences the delivery delay as well as the overhead on transport layer, a careful parameterization of the coding architecture is crucial. The dynamic configuration must not only consider the dynamics of the network state but also the variable source rate of the media stream. The proposed error control scheme is designed to receive fine-grained parameter updates without affecting the continuity of the delivered real-time media content.

Protocol Design

This thesis is accompanied by the design and implementation of a *Predictably Reliable Real-time Transport* (PRRT) protocol. The protocol wraps around the above adaptive,

packet-level error control scheme and implements the required parameter signaling as well as a minimum header format. The protocol is developed in order to address the aforementioned issues of available transport layer protocols. Specifically, it implements loss-tolerant media transport under delay and reliability constraints. It receives the constraints on reliability and delivery delay via its application interface and observes the underlying network state continuously. Based on this information it performs optimized error control in order to deliver the required QoS to the loss-tolerant real-time application. By design, the transport protocol supports scalable media delivery via multicast. It specifies scalable feedback mechanisms for one-to-many error control and implements a sophisticated *timing model* to synchronize the exchange of packet loss notifications and repair packets within the multicast group.

Performance Model

The interaction of packet-level FEC and the reactive transmission of repair packets in the adaptive error control scheme allows for a large number of potential configurations. Consequently, the level of reliability as well as the overhead generated by a certain configuration of the predictably reliable protocol is not immediately obvious. This thesis develops a *stochastic protocol performance model*. The performance model allows to predict the protocol's residual packet loss rate under the application's delay requirement and the observed network state. Besides, the model returns an estimate of the coding overhead introduced by the configuration of the packet-level error control scheme as well as the header and messaging overhead caused by the transport protocol itself. The model predicts the protocol performance in unicast as well as multicast transmission and allows for the integration of various packet loss and queueing models in order to estimate the network's packet loss process.

Reliability Control

In order to optimally meet the application's QoS constraints, the protocol parameterization must be adapted instantly to variations of the media source and the network state. Based on the above contributions, this thesis formulates a framework for *adaptive reliability control* under delay, reliability and bandwidth constraints. The control circuit relies on receiver feedback about the observed network state. The packet loss behavior of the network path is fed into a *block-erasure model*, while time-related measures such as the round trip delay and the media stream's packet interval update the protocol's *timing model*. The protocol performance model simulates the residual packet loss rate after application of the loss-tolerant coding on the estimated network state. The timing model decides which configuration is feasible under the application's delay constraint. Based on this information the control circuit adjusts the protocol parameters for the upcoming observation phase. The reliability control supports *equation-based congestion control*. A suitable rate control equation determines a bandwidth constraint for the protocol performance model.

1.2. Internet Reliability - Related Work

Various concepts contribute to the transport reliability at different positions within the Internet protocol stack. Two fundamentally different types of Internet flows must be

considered in the design of reliable transport mechanisms [15]. Transfer of digital documents, such as files, emails and webpages, is concluded under the term *elastic traffic*. Elastic traffic consumes network resources for a certain duration depending on the file size and the transmission rate under the available network capacity. Multimedia traffic, in contrast, is concluded under the term *inelastic traffic*. Inelastic flows cannot be rate-controlled, since they are offered by real-time multimedia applications that formulate a specific transmission rate throughout their lifetime. Similarly, the provision of reliability is difficult for such flows since most error control schemes introduce variable throughput and transmission delay.

Both aspects are barely addressed in the design of available transport protocols. Yet they differentiate between *total reliability*[3] and *partial reliability*. Elastic traffic requires total reliability, i. e. zero residual transport error must be guaranteed. Intuitively, totally reliable error control causes variable delivery delay as the scheme applies a large number of transmission retries depending on the network's packet loss rate. This property renders suchlike error control infeasible for inelastic flows. Partially reliable protocols are therefore developed to provide such flows a limited delivery delay under variable reliability.

Queueing losses represent the prevalent error source in wide area networks. Traffic engineering and flow management on the network layer can support the reliability and the efficiency of transport layer protocols by implicitly eliminating this error source. Managed QoS frameworks such as the widely applied IP Multimedia Subsystem (IMS)[4] consist of a comprehensive collection of standardized protocols for session management and resource reservation, which is spanning over several layers of the Internet's protocol stack in order to improve the reliability of inelastic flows.

1.2.1. Reliable Transport

A reliable transport protocol hides the packet loss dynamics of the underlying network architecture from the distributed application by ensuring integrity of the data as well as ordered delivery. Different protocols on different layers of the OSI network model (Open Systems Interconnection Reference Model) contribute to reliable data transport over IP networks. In general, transport layer is the uppermost layer to provide reliability. It is also the lowest layer to provide total reliability.

IP networks enable the complete set of error recovery techniques since they rely on bidirectional communication. This allows for *reactive* as well as *proactive error control* schemes. On wired Internet paths, packet loss due to physical transmission errors is rare. Packets are frequently lost due to saturated queues at intermediate nodes that are not able to service the aggregate arrival rate. Therefore, reliable protocols usually implement two components: *error control* and *congestion control*.

The following sections summarize essential ingredients and concepts of reliable transport protocols. Error control and congestion control are closed-loop protocol functions that require regular feedback from the receiver. This is conveniently obtained in a point-to-point topology but arbitrarily challenging in presence of large receiver groups. Hence, substantial research has been carried out in the field of reliable multicast.

[3] The error control scheme is able to repair all detected errors. Error detection, however, relies usually on cyclic redundancy check, which is not totally exhaustive.

[4] http://www.ibm.com/developerworks/webservices/library/ws-soa-ipmultisub1/

Error Control

Reactive packet recovery is widely applied in transport layer error control. In the configuration of an automatic repeat request (ARQ), data segments are repeated by the sender until being acknowledged by the receiver. Retransmissions are initiated upon a timeout at the sender. The receiver omits the acknowledgment in case the segment contains corrupted bits, is completely lost or has been reordered beyond a specific depth. Integrity check, e. g. via cyclic redundancy check (CRC) [105] as well as sequence number check are essential components of reactive error repair. In order to increase the scalability of reactive schemes, the policy of communication can be inverted: The deployment of negative acknowledgments reduces receiver feedback to explicit loss notifications.

Proactive error control via FEC has not been considered for integration into transport layer protocols for a long time since it adds computational as well as architectural complexity. Nevertheless, this technique can compensate for the fundamental drawbacks of reactive error control: FEC significantly increases the scalability of reliable protocols since it requires less feedback and corrects spatially uncorrelated errors with high efficiency. As the OSI layer model requires the integrity of the data forwarded to higher layers, transmission errors result in the loss of entire packets such that packet-level coding on transport layer is equivalent to *erasure coding*. Algebraic maximum distance separable (MDS) codes [133] have the desired property of recovering the entire message from any k out of $n > k$ coded packets. Byte-wise interleaving techniques ensure that each code word just contains a single byte from each packet. Packet-level block-erasure codes based on the Vandermonde matrix [152] have been extensively studied by Rizzo [129]. Vandermonde erasure codes render erasure coding feasible on computer systems even at high data rates.

The intrinsic objective of error control and code design is to approach Shannon's theoretical throughput limit [139] as closely as possible by increasing the code word length. Rate-less erasure codes have been found to significantly reduce the coding and decoding complexity, especially for very long code words (preferably longer than 10000 symbols) [100]. The code generates a virtually unlimited number of coded symbols out of a set of k source symbols based on a pseudo-random degree distribution, which leads to their classification as fountain codes. Since decoding success is no longer deterministic for those randomized codes, an asymptotically constant number of excess symbols η needs to be transmitted in order to recover the original message. Other than the MDS codes, fountain codes require $k + \eta$ symbols to be transmitted in order to recover k source symbols in absence of symbol erasures. η amortizes well for huge code word lengths but causes significant overhead for short code lengths. The efficiency of rate-less codes has been improved under concatenation with deterministic erasure codes. This concept has been applied in the design of Raptor Codes [142].

Reactive error control is inherently adaptive since redundant data are only sent upon the observation of transmission errors. Besides, it is the only technique to achieve total reliability. However, since every repair cycle requires bidirectional communication between sender and receiver, this comes at the price of unpredictable delay. FEC, on the other hand, is inefficient under dynamic network conditions if it is applied with static configuration, but it enables error control under time constraints. Since both schemes mutually compensate for each other's drawbacks, they have been combined within so-called hybrid ARQ (H-ARQ) or hybrid error coding schemes [127].

Congestion Control

Congestion results from bandwidth utilization beyond the network path's capacity. An overloaded network produces clustered packet erasures that affect the overall reliability. Therefore, congestion control is considered to be an error avoidance scheme, preserving the network's stability. The first congestion control algorithms for transport layer protocols have been suggested by Van Jacobson after a couple of congestion collapses experienced in the Internet [79]. Congestion control has mainly two objectives: First, network overload should be avoided, while second, each protocol flow should obtain a fair share of the available bandwidth. Hence, congestion control is a distributed multiple access scheme.

A flow's throughput is controlled via the size of its congestion window, which determines the maximum number of packets or segments that may be offered to the network within one round trip time, i. e. the time between sending a segment and receiving the corresponding acknowledgment. In general, congestion control applies event-driven algorithms in order to adapt the congestion window. Within the last decade, numerous control schemes have been described by stochastic models [68, 31, 116] and many recent developments rely on purely equation-based design [83, 160, 29, 58]. Equation-based congestion control obtains the temporally variable protocol throughput as an explicit result of a control equation instead of maintaining a congestion window. This allows congestion control to be conveniently formulated as a throughput or network utility maximization (NUM) problem, which understands reliable network throughput as a trade-off between rate pricing and application utility [168].

Loss-based congestion control considers packet loss as a congestion notification signalizing queue overflow at some position of the network path. However, loss-based indicators suffer from a high ratio of false positives on wireless segments, where packet loss derives rather from the corruption of single bits than from queue overflow. In order to differentiate between corruption and congestion losses, the one-way or round trip delay can be observed, whereas an increasing delay is interpreted as an indicator for a rising queue level [115, 27, 37]. Similarly, explicit congestion notifications (ECN) can support the loss differentiation by marking packets in case of a critical queue level [6, 89].

Elastic data traffic is well suited to the use of closed-loop congestion control [63]. A congestion control algorithm instantly probes the available bandwidth by increasing the throughput until a congestion event requires a window reduction. A widely applied policy is additive increase and multiplicative decrease (AIMD), whereas other functions, such as logarithmic [29] or cubic equations [68], might describe the window growth.

Non-elastic multimedia traffic renders both congestion control as well as erasure recovery techniques difficult. If running in parallel to AIMD-controlled flows, non-elastic flows suffer from packet loss since loss-based AIMD congestion control induces small congestion events in order to probe for available bandwidth. However, during the congestion event it is counterproductive to add redundancy to the inelastic flow since it increases the saturation of the network. Therefore, especially the combination of error and congestion control of inelastic flows is a challenging research topic. Some work has been done on the prediction of congestion events or on the development of learning algorithms in order to anticipate congestion loss with proactive repair packets [136, 57, 130].

Point-to-point Reliability

Following the Internet's end-to-end principle, transport layer reliability is generally implemented in the end hosts without support of intermediate nodes. The Transmission Control Protocol [125] is the de facto standard for transport reliability in the Internet providing both congestion control and error control. TCP implements a reliable byte stream like a remote pipe between two hosts, which is the basis for many distributed applications without error tolerance.

TCP formulates a delivery guarantee for each byte via sequence numbering. The sender socket expects each sequence number to be acknowledged. In case of out-of-order transmission, damaged or erased segments, the acknowledgment contains the last correctly received byte's sequence number. TCP has been enhanced by a fast retransmit option that avoids waiting for a timeout in case the receiver recognizes an erroneous transmission. It optionally implements selective acknowledgments (SACK) such that not the whole window but just the missing segments are repeated in case of an error. The protocol is connection-oriented by implementing a handshake algorithm and the connection state is tracked by a state machine.

While TCP's error control has basically remained unchanged to date, intensive research has been carried out on improving and customizing its congestion control for different demands and environments. Standard TCP applies window-based AIMD congestion control, which is implemented within various flavors that are optimized for different applications and network infrastructures. New TCP flavors have mainly been proposed in order to address TCP's throughput inefficiencies on network paths with large bandwidth-delay product [83, 68, 160, 97, 148] and wireless networks with high packet loss rates [150, 101].

Besides TCP, few reliable point-to-point protocols have been developed, usually implementing small customizations unrelated to the congestion control algorithm. The Reliable Data Protocol (RDP) [117] implements in-order delivery as an optional component that allows for higher throughput if it is disabled. It was mainly designed for reliable bulk data transport. In contrast to TCP, it preserves packet boundaries as it operates on datagrams. Similarly, NETBLT [40] has been proposed as a reliable datagram service with increased throughput.

The Stream Control Transmission Protocol (SCTP) [143] is a recent approach to address TCP's shortcomings, in particular for the transmission of telephony signaling. SCTP specifies aggregate TCP-like connections that implement in-order delivery, congestion control and error control separately for each stream. In addition, urgent messages can be transmitted out of order. SCTP supports multi-homing and multi-streaming if the end hosts have access to several networks. SCTP performs chunked data transport that is similar to a datagram service. The connection status is monitored via keep-alive messages.

The integration of FEC into TCP has been evaluated in order to improve TCP's robustness against physical packet loss. The scheme has been implemented into loss-tolerant TCP (LT-TCP) [150]. LT-TCP differentiates between congestion and corruption losses by exploiting ECN. Hence, in case a segment loss appears within an apparently uncongested network path, LT-TCP includes FEC parity segments proactively and reactively into the congestion window. In addition, TCP's maximum segment size (MSS) is tuned to enable fine-grained control over the FEC.

Reliable Multicast

IP multicast implements the most scalable and efficient data delivery to large receiver groups based on the assistance of intermediate network nodes. Unfortunately, multicast support is still incomplete in today's Internet due to the optional implementation in IPv4 [124] and its impact in the near future is still hardly predictable. The deployment of IPv6 [48] is considered to be a driving factor for the establishment of Internet multicast services. Importantly, it promises gapless support of multicast routing as this is a mandatory feature in the standard and it removes a fundamental barrier for multicast communication by rendering network address translation obsolete. Despite the significantly less efficient resource utilization, most of today's one-to-many services still prefer to scale hardware and network infrastructure in order to maintain multiple point-to-point instead of point-to-multipoint connections to larger groups of receivers.

Both error control and congestion control are particularly challenging in multicast. An inherent problem is the potential heterogeneity of the receiver group with respect to loss, delay and available bandwidth, which is a common problem due to the presence of wireless segments in the network path to the consumer. Any reliability scheme designed for multicast is therefore subject to the problem of finding a good representative for the reception quality of the entire group. This representative is usually a virtual receiver composed of worst-case measurement samples retrieved from the group. Additional communication overhead is generated to determine the representative.

Scalable reliability for multicast groups has been extensively studied within the last two decades. Common problem of multicast error control is the fact that receiver feedback scales linearly with the group size. So-called *feedback implosions* may either saturate the network close to the multicast source or might overwhelm the centralized error control scheme [164, 110]. Therefore, scalable error control requires support from the receivers as it clearly does not comply with the idea of multicast if the sender has to maintain knowledge about the whole group [60]. For instance, negative acknowledgments (NACK) reduce the feedback load significantly at the cost of increased complexity at the receiver as it has to maintain transmission timers [22, 121].

Reliable multicast has been a major motivation to integrate FEC into network protocols since it implements perfectly scalable error recovery. However, without proper adaptation to the dynamic reception quality, it wastes a considerable amount of bandwidth. Adaptive FEC requires stochastic modeling of the channel behavior [111], whereas timely adaptivity is considered to be an open issue [132].

The problem of feedback implosions has been addressed within several reliable multicast approaches by various feedback suppression techniques [110]. The Reliable Multicast Transport Protocol (RMTP) implements tree-based multicast, which separates multicast groups into regions represented by one designated receiver that serves local retransmissions [118]. As a result, the feedback of each region is aggregated at the designated receiver that in turn is reliably connected to the multicast source.

The Xpress Transport Protocol (XTP) [159] implements reliable transport either point-to-point or one-to-many. The receivers use a slotting technique to send their negative acknowledgments with random delay in order to avoid feedback implosion. Other than that, periodic polling schemes have been proposed to make feedback scalable for error and flow control [13]. Some schemes rely on the support of intermediate nodes (router assisted) in that they suppress redundant NACKs [164].

A comprehensive protocol suite for reliable multicast has been specified as NACK-

oriented Reliable Multicast (NORM) [3]. NORM is work in progress. It implements both, multicast congestion control and error control as well as feedback suppression. The protocol combines proactive as well as reactive error control in variable configuration. Reactive error recovery is obtained via selective ARQ based on negative acknowledgments.

1.2.2. Partial Reliability

Bounded transport delay and total reliability cannot be guaranteed at the same time. For instance, a TCP sender introduces arbitrary delay by repeating a data segment until it receives the corresponding acknowledgment. This property renders it infeasible for interactive applications such as telephony and video conferencing services. Since such services require timely data delivery under tolerance for some residual errors, various partially reliable protocols and protocol extensions have been developed.

Partial reliability relaxes the reliability constraint – sometimes also the requirement for ordered delivery – in favor of reduced transmission delay. A common policy is the truncation of repetition cycles for reactive error correction. Similarly, FEC without retransmission opportunities is partially reliable since it is an open-loop error control scheme. Other than total reliability, implementations of partial reliability operate either on link layer, transport layer or even application layer. Partial reliability might also result from the cooperation of several of those layers.

Application Layer

Partial reliability is frequently built upon the basic datagram service provided by the User Datagram Protocol (UDP) [123]. UDP implements a minimal protocol header for source and sink port, datagram length and cyclic redundancy check in order to ensure packet integrity. Therefore, many partially reliable protocols have been implemented on application layer based on UDP's port multiplexing and error detection capabilities.

The Real-time Transport Protocol (RTP)[135] adds a header of 12 bytes to the transport layer protocol, which is usually UDP. Main features of the RTP header are packet timestamp, sequence number and synchronization source identifier. RTP is particularly designed for telephony and video conferencing applications. The additional header fields allow the receiving host to remove packet jitter, restore packet ordering and to multiplex media streams from different sources. QoS provisioning is therefore shifted to the end hosts that evaluate the additional header information in order to improve the rendering quality.

RTP can achieve partial ordering by sufficiently large reordering buffers at the receiver, but it does not care for reliability. In general, RTP sessions are accompanied by the Real-time Control Protocol (RTCP) [135]. RTCP contributes a canonical naming service, the dissemination of feedback on the reception quality via receiver reports as well as sender reports that conclude sending statistics. Several RTP profiles have been standardized in order to define the interaction of RTP and RTCP. Specifically, RTP/AVPF (audiovisual profile with feedback) [114] implements partial reliability via a limited number of transmission retries based on RTP/RTCP. RTP/FEC (general forward error coding)[92] is partially reliable by adding packet-level FEC.

RTP/AVPF allows a receiver to notify packet loss within the receiver report by appending a negative acknowledgment block including the sequence number of the lost packet and a bit map for the 16 following packets. RTCP strictly limits the interval

for receiver feedback to a minimum of five seconds in order to preserve the scalability. Therefore, RTP/AVPF comes with an early packet option that allows the receiver to send its receiver report earlier within the minimum interval. This option reduces the delay required for error coding, whereas just a small ratio of errors can be corrected due to the considerably large feedback interval.

Via the RTP/FEC profile, RTP packets can be protected with packet-level block-erasure codes. The profile specifies a virtual interleaving scheme that includes exactly one byte from each packet within each code word. An additional header of 12 bytes is added to the resulting redundancy packets in order to recover the original header fields of erased RTP packets. The parity header communicates the block coding parameters to the receiver and includes a base sequence number such that the receiver can identify the packets belonging to the coding block.

Partial reliability has been implemented on application layer within several reliable multicast protocols since it allows to handle receiver heterogeneity with adjustable overhead. Log-based Receiver-reliable Multicast (LBRM) [75] introduces logging servers in order to provide receiver-based reliability. The logging servers maintain a backlog of the transmitted packets in order to serve receiver-initiated retransmissions asynchronously to the original data transmission. Distributed logging servers keep retransmissions local and improve the protocol's scalability. Similarly, the Structure-Oriented Reliable Multicast protocol (STORM) [165] keeps packet retransmissions local via a hierarchy of multicast nodes. In general, each node is assigned a number of parent nodes that are supposed to answer the node's retransmission requests. Retransmission requests that cannot be answered before a specific deadline are discarded.

A partially reliable real-time multicast scheme has been implemented and evaluated by Rubenstein [132, 131]. Specifically, this scheme combines proactive and reactive sending of FEC parity packets. Proactive packets reduce the protocol's transmission delay and the negative receiver feedback. The protocol tries to meet an application-specific reliability level by adjusting the amount of proactive parity while minimizing the number of reactive transmission rounds.

Transport Layer

Few partial reliability schemes have been implemented on transport layer. The transport-layer approaches follow mainly two design concepts: Either partial reliability, in-order delivery and congestion control are added to UDP in a bottom-up approach, or TCP-like protocols are modified in order to accept delivery deadlines by limiting the number of transmission retries.

Since UDP transport does not implement congestion control, it acquires network bandwidth aggressively and affects fairness and stability within the network. The Datagram Congestion Control Protocol (DCCP) [87] has been designed to provide TCP-friendly congestion control without reliability for interactive datagram services that prefer timeliness over reliability. DCCP implements positive acknowledgments to support the congestion control. Therefore, the protocol is a convenient basis for partial reliability extensions. DCCP with partial reliability (PR-DCCP) [90] differentiates between packets requiring reliable and unreliable delivery. For instance, key frames in a video streaming session should be transmitted with reliability requirement, whereas predicted frames might use the option of unreliable transport in order to minimize the overall video distortion.

Reliable UDP (RUDP) [151] is an early approach of providing in-order delivery and a limited number of packet repetitions for loss recovery. The motivation behind RUDP is the lack of a reliable transport mechanism for telecommunication signaling protocols without significant delay. In particular, these signaling protocols prefer a reliable datagram service instead of TCP's byte stream. RUDP is a connection-oriented protocol that implements TCP-friendly congestion control and flow control.

Time-Lined TCP (TLTCP) [107] modifies TCP in order to assign deadlines to data segments. TLTCP operates similar to TCP until the deadline for a segment expires. Outdated segments are dropped in favor of new data. This allows TLTCP to perform partially reliable error control and TCP-friendly congestion control for multimedia streaming applications.

The SCTP has also been extended by a partial reliability option (PR-SCTP) [128]. Similarly to PR-DCCP the PR-SCTP sender can individually decide for each message whether it has to be delivered with or without reliability. In addition, it can virtually assign a reception deadline to each message via a specific control packet that tells the receiver to increase the sequence counter despite the fact that some missing chunks have not been received so far. This leads the receiver to discard outdated chunks.

Link-layer and Cross-layer Approaches

In addition to protocol enhancements, suggestions have been made to deal with delay and reliability constraints on lower layers. These approaches have the common objective of avoiding or partially correcting packet erasures at lower layers in order to improve the protocol performance on transport layer. As a result, reliable protocols such as TCP observe a reduced loss rate and improve significantly in throughput. In some cases the schemes provide already sufficient reliability on lower layers such that they render transport layer reliability obsolete. Due to the fact that significant exchange of information between several OSI layers is required, they are considered to be *cross-layer schemes*. Yuksel et al. [171] proposed to inform the IP layer about the link layer loss rate in order to initiate a fast failover to alternative routes in case of high loss rates. The scheme is specifically designed to support the multicast of IPTV services over UDP and RTP. It complements the application of multicast reliability on transport layer, which is subject to scalability problems.

Reliable multicast is particularly challenging in wireless networks. Therefore, customized schemes have been developed for different wireless IP standards in order to support partially reliable multicast. In the IEEE 802.11 MAC (medium access control) layer, so-called leader-based protocols achieve high correction performance under low delay and small overhead [95]. Nevertheless, it is recommended to combine them with a transport or application layer scheme in order to deal with the residual error floor. Several cross-layer coding schemes on IEEE 802.11 networks have been evaluated by van der Schaar et al. [154].

The Autonomic Transport Framework developed by the LAAS Toulouse optimizes available transport and QoS allocation methods to offer partial order and partial reliability [55]. The framework specifies an implicit packet meta header that enables cross-layer communication of QoS requirements. For instance, a media stream's delay and reliability constraints could be made available to all layers in the multimedia communication stack in order for them to undertake respective actions to meet those constraints.

1.2.3. Managed Internet

The simultaneous transmission of elastic as well as inelastic, audiovisual flows over the same network infrastructure renders self-organizing protocol behavior difficult. This is a result of their fundamentally different requirements. For example, inelastic flows compete with elastic flows for available bandwidth in an unfair fashion. On the other hand, elastic flows, mostly controlled by the TCP, affect the QoS of audiovisual traffic by inducing significant loss rates as a result of their loss-based AIMD congestion control.

Even though the QoS requirements of voice and video sessions have clearly been defined within the ITU-T Y.1541 QoS classes [137], the standard socket interfaces of available transport protocols lack the support of communication mechanisms in order to configure and support QoS guarantees. It is believed that suchlike functionality must be controlled from an overall control framework [9]. Such control frameworks conclude transport and signaling protocols as well as admission control and traffic engineering in order to provide managed end-to-end transport of multimedia flows while guaranteeing their QoS requirements. Managed Internet does not explicitly provide reliability. It rather implements error avoidance by careful resource reservation and over-provisioning.

Resource Reservation

In the past, sophisticated queueing models have been proposed to model different arrival characteristics of various types of Internet traffic [19]. The models also describe the way this traffic is serviced at intermediate Internet nodes. Queueing models are the basis of Internet traffic engineering, i. e. the proper adjustment of queue lengths and flow shapers at forwarding Internet nodes in order to multiplex arriving traffic at the QoS it requires.

Resource reservation is often characterized to be open-loop congestion control as it assures a certain share of network bandwidth to a flow. It adds significant overhead since dedicated protocols are used to initiate and update the reservations at the routers along the network path. Reservations are only valid if they are periodically being updated. Otherwise they expire in order to reset reservations owned by receivers that left the network unexpectedly. Resource reservation is executed at the forwarding nodes by installation of traffic shapers or by application of special scheduling algorithms in order to prioritize traffic with strict QoS requirements. Substantial over-provisioning in bandwidth reservation is necessary to handle variable bit rate streams.

Admission control is applied to avoid network saturation and congestion losses in managed networks. A bandwidth broker manages the bandwidth shares for admitted flows. Before the flow is admitted to obtain a share of the network resource, the bandwidth broker compares its QoS requirements with the available resources. In case sufficient resources are available, the flow is admitted and the resource reservation is initiated via the respective signaling protocols.

The Internet Engineering Task Force (IETF) proposed two architectures to manage Internet flows. Integrated Services (IntServ) [39] have been standardized in order to provide signaled QoS, i. e. end hosts communicate their QoS requirements at the time the connection is established. Consequently, IntServ requires to maintain per flow states at each router in which the respective resource reservation is stored. The Resource Reservation Protocol (RSVP) [24] is used to update flow states at the intermediate nodes. Other than that, DiffServ [109] relies on packet marking using the IP Type of Service (ToS) field. The packet marker identifies the packet's QoS requirements, i. e. its Class

of Service (CoS). Routers are set up to handle the packets according to the requirements of their respective CoS.

QoS Management

Currently, high-quality video services such as IPTV cannot be delivered over unmanaged Internet. None of the available transport protocols can guarantee sufficient QoS. Therefore, such services are exclusively distributed within the Internet Service Provider's (ISP) own infrastructure. Within this scope the ISP is able to manage the multimedia flow's QoS. The QoS requirements in terms of average and peak bandwidth, maximum jitter and loss as well as delay constraints are formulated within a service level agreement (SLA) between customer and ISP. Basically, the level of service is the certainty of meeting these QoS requirements. The ISP ensures the SLA via the corresponding resource reservation.

An example for suchlike call bandwidth reservation is the IP Multimedia Subsystem. IMS is a complete QoS framework spanning over heterogeneous networks, including wired, wireless and mobile infrastructure. It has been developed under the 3rd Generation Partnership Project (3GPP) and aims in particular at the convergence of cellular and Internet technology. IMS is widely used for voice and telepresence services. It applies the Session Initiation Protocol (SIP) in order to set up communication channels. As a part of the setup procedure, IMS negotiates the end-to-end QoS provisioning for the respective session.

1.3. Predictable Reliability

Managed resource allocation is today's single option to predictably protect continuous media streams from queueing losses and delays on Internet paths. Consequently, commercial voice and video communications services rely on call bandwidth reservation, where the QoS they require is guaranteed via service level agreements. However, many network segments are not under the network service provider's control. This refers especially to scenarios in which the multimedia traffic is routed beyond the borders of the managed infrastructure or over a wireless home network segment.

Without explicit management, packet-switching Internet is a best-effort service implementing reliability on transport layer. Available transport layer protocols observe the network state along the whole end-to-end network path but they do not offer any interface to communicate and negotiate QoS requirements [9]. In addition, adequate path monitoring techniques that are necessary to verify these requirements are missing within those implementations.

Suppose a transport protocol that allows the application to formulate QoS requirements in terms of delay, reliability and bandwidth. Since these parameters are not mutually independent on an unmanaged Internet infrastructure, the protocol has two main features: It establishes a virtually managed pipe through the unmanaged network that satisfies the application's requirements at high probability while managing bandwidth allocation and reliability in a self-controlled fashion. In addition, it instantly monitors the path's quality and notifies the application if the service cannot be delivered under the formulated requirements.

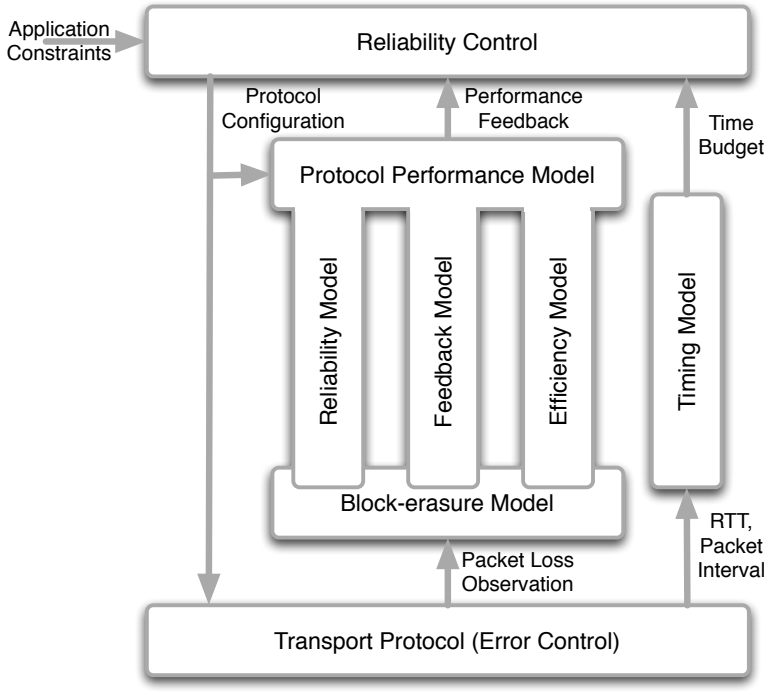


Figure 1.1.: Predictable Reliability – overall architecture

This thesis defines ***Predictable Reliability****, which describes a first approach to apply adaptive error and resource control within a transport layer protocol so as to meet the QoS requirements of multimedia applications.*

Predictable reliability founds on several recent protocol design patterns [31], such as *reliability control* and *equation-based congestion control*, *stochastic modeling* of the *network state* and the *protocol performance* as well as *modular protocol architecture*. In the following subsections, a high-level architecture of the predictably reliable protocol design is being presented. Each component of the architecture is being discussed in more detail within the remainder of the thesis.

1.3.1. Architecture

Predictably reliable transport assigns a *delivery time budget* to every single packet. The budget is specified between the time the packet enters the protocol stack at the sender and the time it becomes available to the receiver application. The protocol must finalize the error coding and repair operations for each packet within its individual time budget.

Basis of the predictably reliable transport is a capacity-approaching *transport protocol* with adjustable, delay-constrained error control (Chapter 2). The protocol observes the dynamic network state of the end-to-end transport path. The information is fed into the protocol's *timing model* as well as a *packet-level block-erasure model* that simulates the network's current packet loss characteristics (Chapter 3). A stochastic *protocol performance model* (Chapter 4) is able to evaluate the performance of the protocol's error control under the simulated network state. The combination of network and protocol modeling enables the protocol's *reliability control* (Chapter 5) to optimize protocol

parameters under the application's QoS constraints. The reliability control regularly configures the adjustable error control with optimized protocol parameters. The overall architecture is presented in Figure 1.1.

Transport Protocol

The transport protocol is the executive unit of the predictably reliable architecture. It implements a scalable (multicast) error control scheme that is optimized under time constraints. The error control is highly dynamic and allows for fine-grained parameter adjustments. In order to support the adaptive error control, the protocol implements receiver feedback within a configurable period and upon packet loss.

Besides the predictably reliable datagram transfer, the protocol specifies a header format for coding parameter signaling and receiver synchronization. The parametrization of the error control scheme must be known at the receivers in order to perform the recovery of lost packets or to request repair information from the sender. Host synchronization is required to coordinate the protocol state and the protocol actions in a multicast group. This applies particularly to the scheduling of receiver feedback. Therefore, a detailed timing model describes the impact of time-related system parameters, such as the packet interval of the real-time source and the network's round trip time. It models the time that is allocated for the error control scheme and incorporates communication delays between the hosts. An overall delay equation for the protocol is formulated.

The transport protocol observes packet loss and delay on the end-to-end network path. It feeds the block-erasure model with measurement samples of the packet loss rate and the pattern of packet losses and sends samples of the round trip time to the timing model.

Block-erasure Model

The packet loss process is potentially observed with a specific burstiness depending on the error source. For instance, queueing losses tend to affect several packets in sequence, whereas a noisy wireless transmission is more likely to produce distributed packet losses. In order to express the burstiness in the packet loss process, a suitable stochastic block-erasure model with memory is being fitted to the pattern of packet losses. The block-erasure model is being updated via statistical evaluation and signal processing on the measurement samples from the protocol's packet loss observations on the network path. The model estimates the network's *packet loss probability*. The accuracy of the model is expressed as a function of the number of available measurement samples.

Chapter 3 discusses a set of potential block-erasure models. Each model generates a specific *block error distribution* that describes the occurrence of errors in a sequence of a certain length. The distribution reflects the packet loss probability estimated from the protocol's loss pattern and it forms the interface between the block-erasure model and the protocol performance model such that the block-erasure model can be replaced without affecting the remaining components of the architecture.

Protocol Performance Model

An essential part of the architecture is the protocol performance model that stochastically formulates the error correction performance of the protocol. Given a set of protocol parameters and the current network state, the model simulates the protocol's residual

packet loss rate and the required coding overhead. Therefore, the module splits into three components, each contributing a specific performance equation.

The block-error distribution provides a probability distribution of the erasure length that is obtained from the block-erasure model. Under consideration of this distribution, the *reliability model* predicts the level of reliability obtained by a given set of coding parameters for the protocol's error control. The model simulates the impact of packet loss on coded source packets. In addition, a *feedback model* determines the effect of missing loss notifications due to packet loss in the return path as well as the result of timer-based feedback suppression in the multicast. The residual packet loss rate of the protocol is formulated as a joint result of both models.

The *efficiency model* determines the protocol overhead of a chosen parameter set based on the simulated network state. This includes the redundancy added by the coding scheme as well as the protocol's header overhead. The efficiency model formulates the expected bandwidth requirement of the protocol for a given source rate.

Reliability Control

The reliability control module adaptively configures the protocol's error control and performs equation-based congestion control. The module formulates predictable reliability as an optimization problem. It receives the delay and reliability constraints from the application and maximizes the protocol goodput under those constraints. In order to optimize the protocol parameters, the module applies the protocol performance model to the simulated network state represented by the timing model and the block-erasure model and obtains the predicted level of reliability. Based on this information it adapts the protocol parameters dynamically to temporally variable network conditions. The optimization problem is solved by a fast search algorithm that benefits from explicit knowledge about the parameter search space. Adaptation of the protocol parameters is performed periodically within a specific interval.

As a result of the parameter optimization, the reliability control module obtains the predicted protocol overhead from the protocol performance model. At the same time the available network bandwidth is obtained from a congestion control equation based on the network state information. Under combination of both information, the reliability control decides whether the desired level of reliability can be achieved under the current conditions and it notifies the application about the feasibility of the QoS requirements under the given network state.

2. Protocol Design

"Without aesthetic, design is either the humdrum repetition of familiar cliches or a wild scramble for novelty."
Paul Rand

Layering is a fundamental design principle of the Internet's protocol stack. Since the OSI layer model keeps the information flow between adjacent layers restricted to small and well-defined interfaces, the stack allows the execution of various transport protocols on top of heterogeneous physical layer implementations. Hence, an application simply defines the communication endpoints of a chosen protocol. However, IP-based multimedia services suffer severely from the lack of suitable interfaces for the communication of their individual transport requirements. The layering even puts a restriction on the transport efficiency [31], which is evidenced by the significant gain in throughput that is achieved by cross-layer transmission schemes, such as application-layer multicast [77], network coding [93] and Internet caching [161].

The realization of QoS allocation and scalability has not received much attention during the development of transport layer protocols. The focus is mainly set on the implementation of total reliability in point-to-point communication, which is required for text-based applications and file transmission. Totally reliable protocols cannot satisfy end-to-end delay constraints because their error control treats single transmission units with unpredictable delivery delay. Partially reliable transport, on the other hand, considers a delivery deadline at the price of an uncontroled residual packet loss rate. Consequently, neither the interface nor the error control specified in available transport layer protocols supports the implementation of predictable reliability.

In order to fulfill the individual requirements of a wide range of real-time communications and interactive services, this chapter presents a novel transport protocol design – the Predictably Reliable Real-time Transport protocol. The chapter develops the protocol specification, including the packet-level block-erasure coding architecture at the heart of the protocol and the corresponding messaging format. The protocol performs proactive and reactive error control according to a defined schedule in order to meet the application's delay constraint. An accurate host synchronization as well as a suitable protocol timing model are being specified in order to execute this schedule in distributed fashion at sender and receiver. The packet-level block-erasure code operates under dynamic configuration in order to process continuous packet streams with variable source rate under a fixed amount of the overall delivery time budget. In order to support the modeling of the network state and the parameter adaptation, the protocol continuously observes the delay and the packet loss behavior of the network path.

2.1. Protocol Specification

The predictably reliable protocol implements the following features and functionalities under consideration of the problem statement formulated in Section 1.1.1:

- **Strict delivery deadline:** Streaming media and interactive services are non-persistent and are absorbed by the sink with a constant delay to the source. Late transmission units are without benefit to the application. The predictably reliable protocol should discard packets perceiving a delivery delay that exceeds the application's rendering deadline. Besides, it terminates the repair process for a source packet as soon as it is obvious that the packet will not be recovered before the deadline. By those means, unnecessary network usage is avoided.

- **Adjustable error control:** The protocol requires a flexible and loss-tolerant error control scheme. In order to meet the delivery deadline, the scheme must allow for parameter optimization under a delay constraint. At the same time, it must provide a controllable residual packet loss rate to achieve predictable reliability.

- **Network monitoring:** The predictably reliable error control relies on a permanent observation of the network state. The network's round trip delay and packet loss rate are of particular interest and must be measured by the transport protocol.

- **Ordered datagram service:** Datagram services are preferred over byte stream services for streaming media delivery, especially in case of loss-tolerant transmission. Since the application is aware of the individual datagrams crossing the network, re-synchronization after the loss of one or several transmission units is possible with the arrival of the next datagram. The protocol must ensure in-order delivery and removal of duplicates.

- **Connection-less communication:** A common principle of media broadcast is the ability of receivers to tune in at any position of the media stream as soon as the reception quality is sufficiently high. Similarly, they might leave a service without explicit teardown of the session. Moreover, in IP-based media delivery session management is performed above transport layer.

- **Scalable transport:** Several media streaming applications such as live media broadcast and video conferencing require scalability from the transport protocol. Therefore, the predictably reliable protocol should support multicast delivery. In particular, error control in multicast requires careful protocol design in order to keep receiver feedback scalable.

The above requirements are reflected within the design of the protocol's packet repair mechanism, the corresponding protocol workflow and the message format as specified within the following subsections.

2.1.1. Adaptive Hybrid Erasure Coding

The information theory does not explicitly include a measure of time. Time consumption of error correcting codes is implicitly fixed by the symbol rate of the communication channel and the code word length. ARQ schemes even introduce a dynamic coding delay

due to their variable number of request and re-transmit cycles. For instance, suppose packet recovery by selective ARQ, where the repair operation either relies on positive or negative receiver feedback. In this case the recovery process of a single packet erasure is likely to require a delay of multiple round trip times.

Bidirectional communication between source and sink is, however, becoming a generic feature of today's media communication systems. It is the key to adaptive channel coding, which allows the source to refine transport parameters according to the receiver's observation of the dynamic network state. For IP media streams the deployment of hybrid erasure coding schemes based on optimal, algebraic block-erasure codes leverages dynamic negotiation of the channel coding rate upon receiver feedback.

Hybrid Error Coding

Packet-level hybrid error coding specifies the sending of coded packets proactively as well as reactively upon receiver request. The joint deployment of transmission retries and FEC provides the flexibility to adapt to a wide range of network conditions with dynamic channel capacity. Under a time constraint, proactive redundancy enables the control over the residual loss rate. FEC brings in the scalability for large receiver groups, whereas the reception feedback from the receivers enables dynamic adaptation of the code rate.

The literature differentiates between three types of HEC. Type-I HEC calculates an erasure code over a block of packets. In case the block-erasure code is not able to recover the block at the receiver, another repetition of the entire coding block is transmitted. This scheme has limited efficiency since packets that have already been received successfully are discarded after each transmission cycle. Type-II HEC improves on that by accumulating correct receptions beyond all transmission cycles. This scheme has been extended by the idea of incremental redundancy, which specifies the sending of a different portion of coded information in each transmission cycle. Finally, Type-III HEC adds per-packet FEC in order to recover corrupted packets from few bit errors instead of discarding them.

In particular, the idea of hybrid error coding (also known under the term hybrid ARQ) has been applied on mobile and wireless network infrastructures. The physical layer of the High Speed Downlink Packet Access (HSDPA) standard for data transmission in third generation mobile networks implements incremental redundancy based on convolutional codes [44]. Type-II HEC schemes have also been considered for efficiency gains in reliable multicast [132, 85], particularly in wireless video multicast [95, 147, 69], while being constructed on packet-level block-erasure codes.

Incremental Redundancy Scheme

Let k be the *number of source packets* handled by the packet-level block-erasure code. The code generates $n-k$ parity packets such that overall n packets exist after the encoding process. Let N_C be the *number of reactive repair cycles* of the HEC scheme. Further, let $N_P = (N_P[0], N_P[1], ..., N_P[N_C])$ be the *repair packet schedule* defining the number of repair packets to be sent in each of the N_C reactive repair cycles, with $N_P[0]$ referring to the initial, *proactive transmission schedule* immediately following the k source packets. Accordingly, the set of $n - k$ parity packets is divided into subsets of size $N_P[c]$, each corresponding to a transmission cycle c with $0 \leq c \leq N_C$ (Figure 2.1).

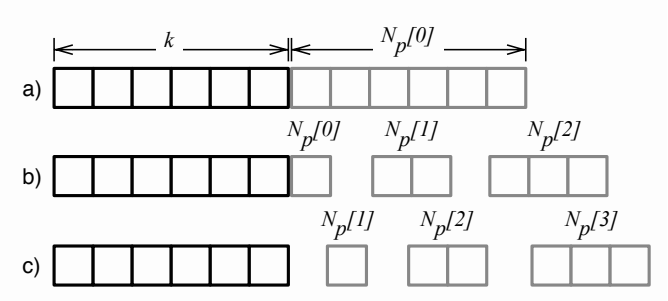


Figure 2.1.: Packet-level hybrid error coding. A Type-II HEC based on packet-level FEC is applied to k source packets. The scheme comprises three modes: FEC mode (a) with just proactive parity, a hybrid mode (b) with proactive as well as incremental (reactive) parity on request and a (purely reactive) incremental redundancy mode (c).

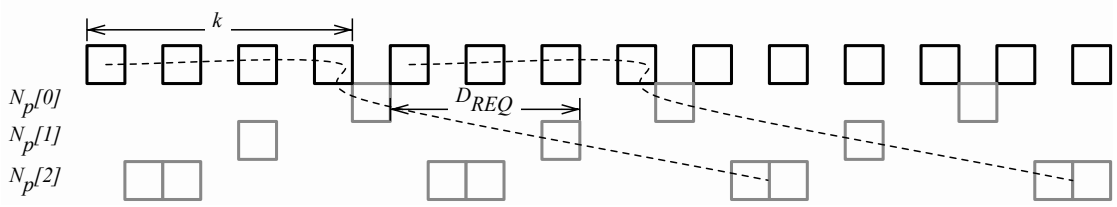


Figure 2.2.: Overlapping repair cycles of subsequent blocks. Repair cycles are scheduled in parallel to the transmission of source packets such that the continuity of the media stream is not affected. Repair cycles of several coding blocks are overlapping if the delay for the collection of k source packets is less than the request timeout for the reactive repair cycles.

The transmission schedules $1 \leq c \leq N_C$ are started after a negative acknowledgment from the receiver, indicating that additional repair packets are required in cycle c. The negative acknowledgments are sent upon the expiration of specific timers at the receiver. The sender immediately answers valid, incoming receiver feedback by transmitting the corresponding amount of parity packets. The receiver accumulates the code symbols received during all cycles in order to reassemble the coded block. Hence, the sender is not obliged to remember individual receiver states and the protocol allows for the implementation in a connectionless fashion, which is beneficial in the multicast.

2.1.2. Protocol Workflow

The combination of block-erasure coding and reactive error repair requires the protocol stack to implement a highly concurrent architecture. Consider the adaptive HEC scheme introduced in Section 2.1.1. The real-time packet stream must be transmitted continuously despite the block-erasure coding. Further, the repair cycles of several adjacent blocks are usually overlapping. This happens if the time required to collect the k source packets of the block is less than the interval between the block's repair cycles D_{REQ} (Figure 2.2). Therefore, the sending of source packets, the process of erasure coding as well as the sending of repair packets must be performed in parallel at the sender. Simi-

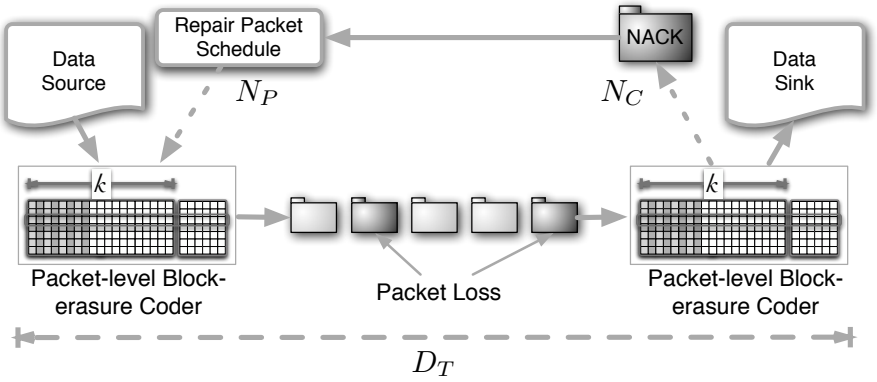


Figure 2.3.: Protocol architecture

larly, the packet reception, the detection and the repair of packet erasures as well as the scheduling of negative acknowledgments are executed concurrently at the receiver.

PRRT's host functionality implements a concurrent architecture of incoming and outgoing packet queues. Packet schedulers for the different protocol functionalities are operating on those queues. Figure 2.3 provides an overview of the distributed protocol architecture. Essential functionalities and data structures of the protocol architecture are introduced in the following.

Encoding and Sending

The sender application continuously inserts source packets into the PRRT socket. The source packets are added into the outgoing packet queue and sent to the network medium as soon as the network interface accepts outgoing transmissions (Figure 2.4, top). The outgoing packet queue stores source packets until they expire, i. e. for a period of the *target delay constraint* D_T that is set by the application.

In parallel, the sender applies packet-level block-erasure coding on a collection of source packets as soon as a sufficient number of new source packets is available in the outgoing queue. For each coding block of k source packets, the encoder constructs $n-k$ parity packets. Larger values of $n-k$ improve the reconstruction capabilities of the code. After the encoding, $N_P[0]$ parity packets are sent immediately.

The sender maintains a repair packet queue that stores either parity packets or source packets. This allows for both sending of coded redundancy packets and retransmission of source packets depending on the protocol configuration. The size of this queue corresponds to the number of source and parity packets collected within D_T.

Error Detection, Decoding and Feedback

The receiver fills received packets into the incoming packet queue (Figure 2.4, bottom). The incoming packet queue provides a data structure for the re-sequencing of packets that are received out of order as well as for the elimination of duplicates. Further, this queue supports error detection and error correction at the receiver. It stores source and parity packets until they expire. At their expiration date, those source packets that have been successfully received or recovered are delivered to the receiver application, whereas

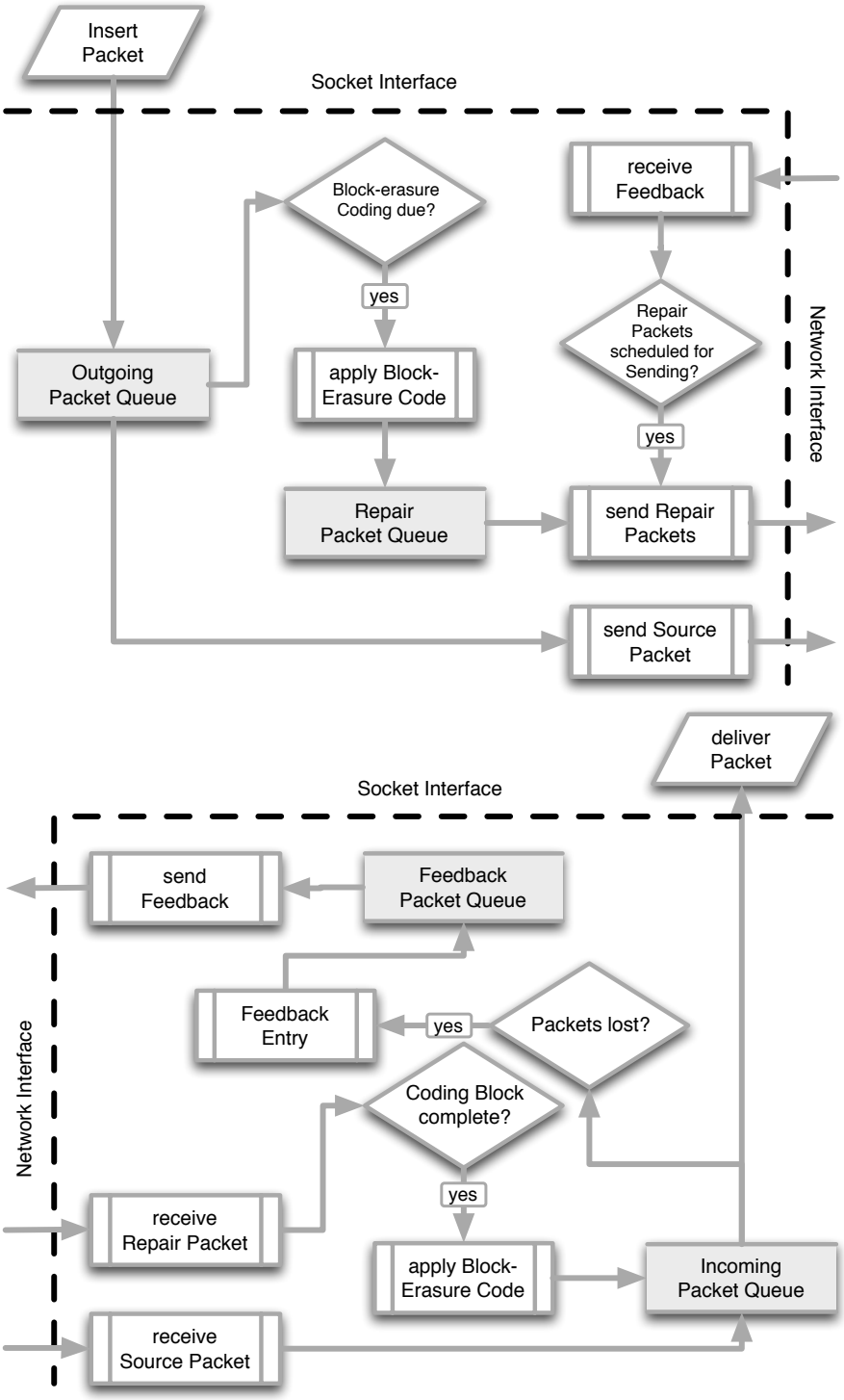


Figure 2.4.: Sender workflow (top) and receiver workflow (bottom).

the related repair packets are deleted.

The receiver detects packet erasures in the incoming packet stream. Upon the detection, a repair request entry with information about lost source packets is stored in the repair request queue. The repair request queue represents the receiver state. It maintains information about the repair state of each independent block.

Incoming parity packets trigger the decoding operation. In case k arbitrary packets from the same coding block are available at the receiver, the decoding operation is successful. As a result of the successful decoding, the recovered source packets are inserted into the incoming packet queue at their designated position. At the same time, all parity packets and the repair request entries belonging to this block are deleted.

If the amount of parity packets is not sufficient to recover the block after a specific timeout, a negative acknowledgment is sent, which notifies the sender about missing packets in the respective block. This process is repeated for each of the N_C repair cycles. Therefore, the feedback scheduler examines the repair request queue periodically and updates the cycle counter of each entry if a new feedback is sent. In case the scheduler observes that a reactive repair trial would exceed the expiration date of the corresponding coding block, no further feedback is sent and the repair entry is deleted from the queue.

Transmission of Repair Packets and Adaptation to the Network State

The adaptive HEC architecture relies on negative acknowledgments for the notification of decoding failure at the receiver. The use of negative acknowledgments in turn requires timeout decisions to be performed at the receiver. As a result, the sender is not required to remember individual receiver states, which improves the scalability of the scheme. In fact, it is sufficient for the PRRT sender to manage global knowledge about the block-erasure coding, specifically, the number of repair cycles spent.

Upon reception of a loss notification, the sender selects repair packets from the repair packet queue. In case the block length is set to $k = 1$, the sender responds to the negative acknowledgment by repeating the original source packet. Otherwise, parity packets are sent according to the transmission schedule N_P.

In order to implement the protocol's temporal adaptivity, the sender evaluates the network state information delivered within the receiver feedback. The network state information includes the network path's packet loss and delay characteristics. Based on this information, the sender updates the timing model and the block-erasure model. In order to keep the models updated in absence of packet loss, the receiver sends periodic network state feedback in a specific time interval.

The sender adapts by continually adjusting the amount of redundancy to the current network state. The amount of redundancy depends on the protocol parameters k and N_P that are optimized based on the evaluation of the protocol performance model given the timing model and the block-erasure model. Modified protocol parameters are immediately applied to the protocol stack. This procedure is implemented by the reliability control module (Chapter 5).

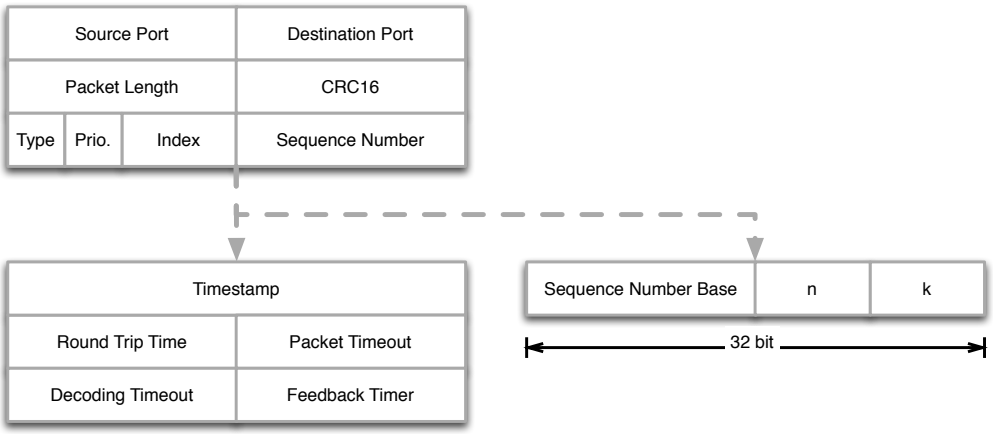


Figure 2.5.: Frame format for source (left) and repair packets (right).

2.1.3. Message Format

General Frame

The protocol inherits UDP's basic datagram service. UDP provides the following header fields [123]:

- **Source Port (16 bit), Destination Port (16 bit):** These fields implement UDP's port multiplexing feature.
- **Packet Length (16 bit):** The length field allows the receiving application to determine the datagram's payload length.
- **CRC16 (16 bit):** UDP applies cyclic redundancy check in order to guarantee the integrity of packets before they are delivered to the application layer. A failed CRC is observed as a packet erasure.

PRRT adopts the UDP header and appends a *general frame* header of 4 bytes in order to multiplex the different packet types and to implement sequence numbering. The general frame header (Figure 2.5) contains the following fields:

- **Type (4 bit):** PRRT's message format differentiates between four packet types:
  - *source packet* (packet type 0)
  - *repetition packet* (packet type 1, repeated source packet for error repair)
  - *parity packet* (packet type 2)
  - *feedback packet* (packet type 3)
- **Priority (4 bit):** This option signalizes the importance level of the packet in case the repair operation implements a prioritization policy.
- **Index (8 bit):** The index field determines the position of source and parity packets within a block of coded packets. The protocol is based on systematic block-erasure codes. The packets from index 0 to $k-1$ are source packets and the parity packets follow at the positions k to $n-1$. This allows the index field to address maximum

256 packet positions, which corresponds to the maximum length of an algebraic block-erasure code operating on byte symbols. The index field resets to 0 for the first packet of every block. Therefore, it indicates the block boundaries to the receiver without knowing the actual coding parameters.

- **Sequence Number (16 bit):** The position of erased packets is detected based on the sequence number. Since the sequence number has a larger wrap-around period than the index field in the general header, it identifies packets uniquely beyond block boundaries and is applied for block addressing. Each packet type instantiates an own sequence number space, which allows to sample the packet loss rate individually for each packet stream. In particular, this field enables the measurement of packet loss in the receiver feedback.

Source Frame

The PRRT *source frame* (Figure 2.5) carries the actual source packets. Since this frame type is sent deterministically and in general with the highest frequency, it facilitates the synchronization of the receiver clock with a reference time sample from the sender clock. The source frame header specifies the decoding timeout and the delivery deadline of the covered payload. It immediately follows the general frame header with the following fields:

- **Timestamp (32 bit):** In order to synchronize the receiver to the sender's time base, a sample of the sender clock is included in each source frame. The sample is taken at the time the packet leaves the outgoing packet queue at the sender. The 32 bit field enables microsecond resolution for the packet timestamps. Based on the timestamp field, the receiver performs jitter cancellation and clock recovery as described in Section 2.2.3.
- **Round Trip Time (16 bit):** This field communicates the sender's current RTT estimate, which enables the receiver to compensate the packet's timestamp for the transmission delay in the clock recovery mechanism (Section 2.2.3).
- **Packet Timeout (16 bit):** The packet timeout specifies the packet's expiration date. The packet should be available to the receiving application no later than the packet timeout.
- **Decoding Timeout (16 bit):** The decoding timeout initiates the reactive part of the packet repair process. It indicates the earliest point in time at which the sender expects receiver feedback for the current packet block, i. e. when the entire coding block should have been collected at the receiver.
- **Feedback Timer (16 bit):** The feedback timer communicates the current interval for the sending of packet repair requests to the receiver. This delay compensates for the communication delay between sender and receiver, the propagation delay of the repair packets, as well as the response latencies at both hosts.

Repair Frame

The *repair frame* (Figure 2.5) is a container for the parity packets generated during the block coding at the sender. The repair header communicates the coding parameters to

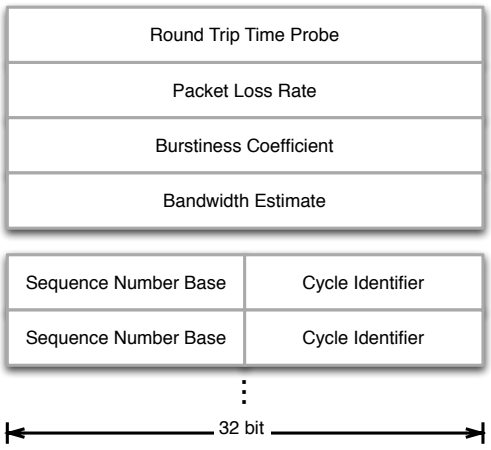


Figure 2.6.: Frame format for receiver feedback.

the receiver. This information is only required upon reception of repair packets in order to prepare the block-erasure decoding at the receiver. A repair frame appends the repair header with the following fields to PRRT's general header:

- **Sequence Number Base (16 bit):** The sequence number base is equal to the first sequence number in the coding block the repair packet belongs to, i. e. it indicates the start of a coding block.
- **n (8 bit):** The code word length n of the block-erasure code.
- **k (8 bit):** The number of source packets k per coding block. For each incoming repair packet, the receiver checks whether k packets out of the block starting with **Sequence Number Base** have been received so far in order to determine whether packet loss recovery for this block is possible.

Under a block length $k = 1$, the sender answers loss notifications from the receiver with the repetition of the original source packets. In this case, the packet type in the general header is set to 1 in order to signal a repetition packet. Otherwise a coded parity packet is sent including the repair frame header. The payload of the repair frame is generated from a block of complete source packets including the source frame header and information about the packet length. This enables the receiver to restore the lost header information.

Receiver Feedback

Feedback packets (Figure 2.6) have dual functionality. Besides *loss notifications* they contribute network state information within the *feedback header*. The sender listens to receiver feedback at the next higher port number with respect to PRRT's destination port. Feedback can be sent either in unicast or multicast. The feedback header is specified as follows:

- **Round Trip Time Probe (32 bit):** RTT probe carries the time stamp of the latest source packet available at the receiver incremented by the time elapsed since the reception of this packet. This information allows the sender to estimate the

receiver's current RTT. Section 2.4.1 details the RTT estimation technique applied in PRRT.

- **Packet Loss Rate (32 bit):** PLR is the packet loss rate estimated by the receiver.
- **Burstiness Coefficient (32 bit):** Provides a metric of the burstiness of the packet loss process observed by the receiver. Depending on the block-erasure model (Section 3.2.2), the value might express the correlation coefficient of subsequent packet transmissions.
- **Bandwidth Estimate (32 bit):** This field denotes the bandwidth that is available for the coded packet stream. It prepares the protocol for rate-based (multicast) congestion control. The field carries the current fair share of the available bandwidth measured by an appropriate congestion control scheme at the receiver.
- **Feedback Block (32 bit):** The payload of the feedback packet identifies the coding blocks that have been received in error. Several feedback blocks can be sent within one feedback packet. A feedback block includes the **Sequence Number Base** of the incomplete coding block as well as the repair cycle identifying the receiver state with respect to that block. The cycle identifier enables the sender to choose the number of repair packets scheduled for the corresponding repair cycle.

In absence of packet loss, feedback headers are sent periodically without loss notifications in order to update the network state information at the sender. In the following this feedback is referred to as *network state feedback*.

2.2. Timing Model

An essential component of predictable reliability is the exact timing in the recovery mechanisms for packet erasures. Given the application's target delay requirement D_T as a time constraint, the predictably reliable protocol must hide variations in the end-to-end delay introduced by the network transport as well as the error recovery scheme by definition of appropriate margins. An accurate timing model is the basis for the optimization of the error control scheme, i. e. the time-sensitive parameters of the scheme must be tuned in order to share the available time budget under variable network conditions.

Since source packets and repair packets that exceed D_T are practically worthless for a real-time application, they lead to residual packet loss. On the other hand, excessive over-provisioning in the time budget reduces the protocol's goodput. Hence, an erroneous timing model may either lead to an increased residual error floor or to suboptimal bandwidth utilization. Different delay sources on the network path follow different characteristics and require separate modeling. The following section differentiates between system-specific delay introduced by the end hosts and the underlying network infrastructure as well as latencies that are immediately determined by the protocol parameters.

2.2.1. System-specific Delay

Transmission and Propagation Delay

The transmission or store-and-forward delay [88] D_{DTx} required to send a data packet of payload length L_D with header length L_{DHdr} through a network with bandwidth

capacity R_C is determined by

$$D_{DTx} = \frac{L_D + L_{DHdr}}{R_C}. \tag{2.1}$$

Let the sending of source packets always be prioritized over the sending of repair packets. Under this assumption and if source packets are sent at a source rate R_S, a remaining bandwidth of $R_C - R_S$ is available for the repair packets. D_{PTx} is defined as the transmission delay of one repair or parity packet with payload length L_P and header length L_{PHdr}:

$$D_{PTx} = \frac{L_P + L_{PHdr}}{R_C - R_S}. \tag{2.2}$$

The transmission delay is independent from the distance between the communicating hosts. The propagation delay, however, is proportional to the distance d by the inverse of the speed of wave propagation p through the transmission medium. D_{PR} is defined as

$$D_{PR} = \frac{d}{p}. \tag{2.3}$$

For a wireless network set $p = c$ where c is the speed of light. A wired network is generally characterized via $0.59c \leq p \leq 0.77c$ [120].

Response Delay

Multitasking at the end hosts introduces scheduling latencies that are referred to as response delays. Response delays occur for instance between the allocation and the sending of a packet or between the recognition of a packet erasure and the sending of the corresponding feedback. For each communication, response delays might occur at both sender and receiver. The aggregate response delays are accounted for with a constant margin D_{RS}.

As an example, Ashvin et al. examined the response time of the Linux kernel under different patterns of background tasks [2]. It has been found that scheduler latencies of up to $20\,ms$ might occur in a preemptable Linux kernel, which reflects the situation in recent kernel versions. For time-critical environments, real-time operating systems have been developed in order to guarantee response times around $1\,ms$.

Queueing Delay

In order to compensate for packets arriving in batches that exceed the service rate, intermediate network nodes perform drop-tail queuing [41]. Depending on the queue level seen by a packet at the time it is being added to the queue, it experiences an individual queueing delay D_Q. The distribution of the queueing delay is strongly dependent on the network utilization as well as the arrival characteristics of the network traffic. Amount and quality of the cross traffic determine the queueing behavior of a network node significantly. Since an error recovery scheme with incremental redundancy sends repair packets in batches as well, the protocol experiences self-induced queueing delay.

The following example should provide the reader an intuitive feeling for queueing delays occurring in the Internet. It has been found by Garetto et al. [62] that TCP

background traffic can be accurately modeled by a $M^X/M/1$[1] queueing model where the superposition of the TCP streams is assumed to produce batches of size X due to their aggregate window size during each simulation step. The length of the TCP sessions has been chosen randomly from a geometric distribution. Experiments and simulations showed that queueing delays of $20\,ms$ happen already with a probability of roughly 10^{-4} at 69% network utilization. For more than 80% utilization, delays of up to $50\,ms$ occur with the same probability.

Proper traffic engineering ensures that queues are allocated in the appropriate sizes, i. e. the queueing delay has an upper limit beyond which packets are blocked at the queue. A common rule of thumb suggests to set the queue size to the bandwidth-delay product of the adjacent link, which is defined by the product of D_{PR} and R_C.

Round Trip Delay

The round trip time is defined as the delay between sending a request to a remote host and receiving the corresponding response. It is determined based on a host's local time base by simply measuring the time elapsed during a bidirectional communication event. The RTT is composed of the propagation delay D_{PR} and the queueing delay D_Q. Under the assumption that those delays are symmetric on the network, the RTT can be modeled as

$$RTT = 2 \cdot (D_{PR} + D_Q)\,. \tag{2.4}$$

In case of synchronized hosts, it is possible to determine the one-way delay or forward trip time (FTT) of a message:

$$FTT = D_{PR} + D_Q. \tag{2.5}$$

Since D_{PR} is usually assumed to be constant for a certain system, the RTT is frequently used to monitor the queue states of the network path via the queueing delay D_Q [160, 130]. FTT provides a more accurate measure as it eliminates the dynamics of the return path and does not require the symmetry assumption.

2.2.2. Feedback Scheduling

The schedule of receiver feedback is determined by two events: First, the receiver needs to determine the point in time at which the sender expects feedback for the first reactive repair cycle. This date is referred to as the *decoding timeout*. Second, after sending a loss notification, the receiver has to wait for the potential arrival of additional repair packets before it repeats the loss notification. Therefore, it sets an adequate *re-request timer*.

It is important for the loss notifications of the receiver to be sent with a deterministic schedule because timing errors in the receiver feedback have significant impact on the protocol performance. The stateless behavior of the sender makes it unable to store receiver feedback. Therefore, an incoming loss notification is answered immediately by the sender if the corresponding repair cycle has not been spent as a response to earlier feedback for the same block coming from another receiver.

The block-internal time measurement starts with the first packet of the block entering the protocol socket. All timers within PRRT are measured relatively to this instant

[1] According to Kendall's notation: a single queue with exponentially distributed inter-arrival and service times, whereas arrivals occur in batches of size X.

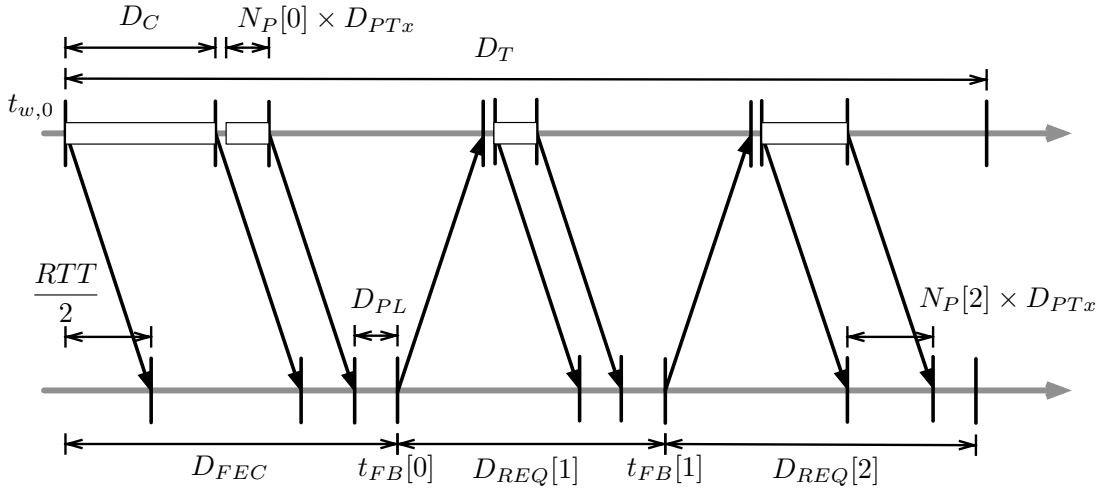


Figure 2.7.: Protocol timing model.

of time. The timers ensure that the first packet of the block is delivered within the end-to-end time constraint. The overall timing model is presented in Figure 2.7.

Decoding Timeout

The sender cannot answer to feedback before the successful completion of the block encoding operation. Storing and delaying of the feedback at the sender until the time it can be answered would affect the scalability of the protocol. Obviously, loss notifications are required not to arrive at the sender before the coding operation is finalized. Moreover, the receiver does not know whether repair packets are already in transit at the time it observes the packet loss. Therefore, the transmission of all k source packets as well as the $N_P[0]$ proactive repair packets must be completed before the receiver decides whether additional redundancy is required for the decoding process.

Let T_S be the current source packet interval. Under the current source rate R_S and the source packet's payload length L_D, the packet interval at the sender is obtained by

$$T_S = \frac{L_D}{R_S}. \tag{2.6}$$

Consequently, the collection of k source packets causes a coding delay of

$$D_C = k \cdot T_s. \tag{2.7}$$

In an unsaturated network, i. e. if R_S does not exceed the available bandwidth, T_S is assumed to be greater than the transmission delay D_{DTx} of the source packets. The coding process is implemented such that it does not delay the source packets at the sender. Therefore, the k source packets are assumed to be available at the receiver no later than

$$D_C + \frac{RTT + D_{RS}}{2} \tag{2.8}$$

time units after insertion of the first packet's first byte, where D_{RS} accounts for response latencies at the sender and the receiver. The repair packets are available after a multiple

of the parity transmission delay D_{PTx} under the assumption that the parity information at the sender is calculated within the response delay D_{RS}. Assume further that the receiver introduces an additional delay D_{PL} in order to detect packet losses in the transmitted block. The overall forward error coding delay D_{FEC} is thus obtained by

$$D_{FEC} = D_C + N_P[0] \cdot D_{PTx} + \frac{RTT + D_{RS}}{2} + D_{PL}. \tag{2.9}$$

Let the margin D_{FEC} be defined in order to compensate for the delay caused by block coding and original transmission. Let $t_{w,0}$ be the date where the first byte of the first source packet is available at the sender. The loss notification that activates the first repair cycle needs to be sent at time

$$t_{FB}[0] = t_{w,0} + D_{FEC}. \tag{2.10}$$

The value is calculated at the sender and communicated in the **decoding timeout** header of each source frame. The header field carries $t_{FB}[0]$ as an offset to the packet's **timestamp** field denoting the time t_s at which the packet is sent to the network medium, i. e. **decoding timeout** $= t_{FB}[0] - t_s = D_{FEC} - (t_s - t_{w,0})$.

Repair Request Timer

After initiation of the reactive repair process by sending the first feedback at $t_{FB}[0]$, the receiver sends additional loss notifications periodically. The period of the receiver feedback is based on two observations: the network path's RTT as well as the transmission delay D_{PTx} of the repair packets sent. A delay margin D_{RS} additionally accounts for the response delay on the communicating hosts. Hence, if $N_P[c]$ repair packets are sent in cycle c, the whole request and repair cycle lasts for a request delay $D_{REQ}[c]$ of

$$D_{REQ}[c] = RTT + N_P[c] \cdot D_{PTx} + D_{RS}, \tag{2.11}$$

where $0 < c \leq N_C$.

Assume that the repair packets scheduled for repair cycle c arrive at the receiver with a request delay $D_{REQ}[c]$ after sending the loss notification for cycle c. Recall that the reactive repair process starts at time $t_{FB}[0]$. Consequently, the feedback for repair cycle $c + 1$ is sent at

$$t_{FB}[c+1] = t_{FB}[0] + \sum_{i=1}^{c} D_{REQ}[c], \tag{2.12}$$

where $0 < c < N_C$. If the receiver cannot recover the coding block at time $t_{FB}[c+1]$, it repeats the loss notification for the block. This process lasts until either a sufficient number of repair packets is received or the remaining time budget is insufficient for the completion of another repair cycle. The overall time budget for each source packet is determined by the **packet timeout** header D_{TO} depending on the delivery delay constraint D_T. If a packet enters the protocol socket at time t_w and is sent at t_s, the packet timeout D_{TO} is calculated as

$$D_{TO} = D_T - (t_s - t_w). \tag{2.13}$$

Each source packet must be ready to leave the receiver's stack at the delivery deadline, which is $t_{w,0} + D_T$ for each packet in the block. Depending on this date, a feedback

deadline exists for the receiver after which additional repair packets cannot be received before the delivery deadline. The receiver must not send additional feedback for cycle $c+1$ if

$$t_{FB}[c+1] > t_{s,0} + D_{TO,0} - D_{REQ}[c+1], \tag{2.14}$$

where $t_{s,0}$ is the **timestamp** field and $D_{TO,0}$ is the **packet timeout** field of the coding block's first packet. In order to simplify the parameter signaling between sender and receiver, a constant timer $D_{REQ} = D_{REQ}[c]$, $0 < c \leq N_C$ is assigned to each repair cycle. The sender communicates D_{REQ} in the source frame header of each source packet via the **feedback timer**.

Effect of Timing Errors

The performance and the efficiency of the predictably reliable protocol depends strongly on the exact implementation of the timing model. However, timing errors might result from delayed transmission units, erroneous receiver synchronization, underestimated RTT as well as excessive processing delay at the end hosts. Four potential cases of erroneous protocol timing are therefore analyzed in the following.

- **Underestimated decoding timeout:** If the absolute timestamp for the decoding of a coding block is determined early, potential proactive repair packets are likely not to have reached the receiver by this point in time. This timing error leads to a premature request of repair packets and might result in two effects. First, repair packets might be sent redundantly to the proactive repair packets upon the early receiver feedback, which leads to excessive coding overhead. Second, the sender might not be able to respond to the repair requests in case the block coding operation has not been finalized until the requests reach the sender. The latter case therefore results in an increased residual loss rate since early repair cycles are potentially being wasted.

- **Overestimated decoding timeout:** Late estimation of the decoding date leads to wasted time budget. The loss of time budget either reduces the coding block length or the number of reactive repair cycles. Reducing the number of repair cycles requires the repair packets to be concentrated on earlier cycles, whereas a shorter coding block length even causes worse amortization of the repair packets in the overall redundancy calculation. The result is a larger coding overhead.

- **Underestimated request timer:** Similarly to an early decoding timeout, this timing error leads to the premature sending of receiver feedback after the previous repair cycle. As a result, additional repair packets are requested before those of the previous cycle might have reached the receiver, which reduces the network utilization.

- **Overestimated request timer:** Overestimation of the repair request timer wastes time budget and the network utilization decreases accordingly.

2.2.3. Peer Synchronization

In general, free-running receiver clocks deviate in phase and frequency since their oscillators are subject to manufacturing tolerances and thermal influences. In the live media

broadcast this problem has been addressed with open loop synchronization mechanisms where the media source periodically sends reference time stamps sampled from its local clock along with the media content [78]. The receivers implement clock recovery algorithms in order to tune their local clock to the sender's reference signal.

In computer networks, dedicated protocols, such as the Network Time Protocol (NTP) [106], the Precision Time Protocol (PTP) [126] and the Global Positioning System (GPS) [45], have been implemented to achieve global clock synchronization. However, whereas NTP and PTP require connection to a specific server that provides the reference clock, GPS is based on satellite reception, where an unobstructed view to the sky is a precondition.

As the access to network synchronization is conditional, PRRT implements an internal synchronization mechanism between the PRRT sender and the group of PRRT receivers. The scheme relies on the principle of clock synchronization as it is applied in the media broadcast and uses PRRT's packet timestamps as a reference clock signal. Instead of adjusting the physical device clock, a virtual clock is maintained at the receiver for each incoming PRRT connection. The proposed algorithm benefits from the presence of global network synchronization via NTP, PTP or GPS in that the system clock of the receiving hosts are globally synchronized. However, it does not necessarily require support from external clock synchronization protocols.

Protocol Time Base

In a multicast environment each PRRT peer maintains two time bases: a *local time base* derived from the own internal system clock as well as a *global time base* synchronized to the multicast source. Therefore, the global time with respect to a specific source is considered to be synchronized among all receivers. On Linux operating systems the system clock is obtained with a granularity of $1\,\mu s$. The protocol's 32 bit Timestamp field in the source frame header carries the resulting $1\,MHz$ timer with a wrap-around period of $4295\,s$, which significantly exceeds any buffering and propagation delays along the network path.

The multicast source determines the global time base via the sampling of its internal clock. The clock samples are included in the timestamp field of the source frame header at the time the packet leaves the sender stack. All receivers listening to the corresponding multicast group tune their local clock based on the reference clock samples.

Each PRRT peer can be multicast source and receiver at the same time by implementing one or several source and sink sockets. Therefore, the protocol timing is implemented locally into the incoming as well as the outgoing packet queue. The outgoing packet queue operates based on the peer's local clock. Each incoming packet queue, however, synchronizes an individual global clock to the reference clock of the corresponding multicast source.

Clock Synchronization

In order to compensate for phase and frequency drift between the sender and the receiver, a digital phase-locked loop (DPLL) is implemented at the receiver, which tunes the local, virtual oscillator to the frequency of the global clock (Figure 2.8). The DPLL circuit removes high-frequency clock jitter from the incoming samples and detects low-frequency clock skew from incoming reference clock samples compared to the predicted clock phase.

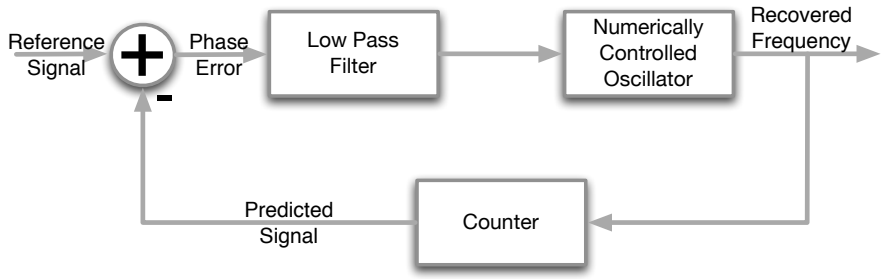


Figure 2.8.: Digital phase-locked loop.

In the following, the functionality of the four essential components of the DPLL – the comparator, the low pass filter, the oscillator and the predictor – are discussed.

The comparator calculates the current clock deviation $\delta(s)$. The amount of the clock deviation depends on the difference between the reference clock sample from the sender and the receiver's virtual clock. The reference clock is delayed by the forward trip time of the the packet carrying the reference time stamp. Let $t_s(s)$ denote the time of sending source frame s and let $FTT(s)$ be the forward trip time experienced by the packet. Further, let $t_r(s)$ be the time of the packet arrival with respect to the receiver's virtual clock. During the transmission between sender and receiver the packet's age increased by $FTT(s)$ such that the reference timestamp t_s is outdated. Therefore, the current clock deviation is obtained as

$$\delta(s) = t_s(s) - t_r(s) + FTT(s). \tag{2.15}$$

The individual propagation delay experienced by each data packet is subject to variations, mainly caused by the queueing dynamics of the Internet path. As a result, the reference time stamps underly high-frequency jitter. Under the assumption of normally distributed inter-arrival jitter, the following recursive low pass filter is applied to the clock deviation $\delta(s)$

$$\bar{\delta}(s) = H(\delta(s)) = \alpha_0\delta(s) + \alpha_1\delta(s-1) + \beta_1\bar{\delta}(s-1), \tag{2.16}$$

where $\delta(s-1)$ denotes the previous sample of the clock deviation and where $\bar{\delta}(s-1)$ is the previous filter output.

This filter equation corresponds to a second order recursive filter with the transfer function $H(z) = \frac{\alpha_0+\alpha_1 z^{-1}}{1-\beta_1 z^{-1}}$ and with the coefficients α_0, α_1 and β_1 [72]. The filtered output is fed into a numerically controlled oscillator (NCO). Under zero input the NCO is assumed to generate a nominal frequency f_n, which is determined by the granularity of the system clock, i. e. $f_n = 1\,MHz$ on Linux operating systems. Otherwise, it increments the output frequency $f(s)$ as a linear function of the filtered clock deviation $\bar{\delta}(s)$ and the sampling frequency. Since a reference time stamp is included in every source packet, the systems sampling frequency corresponds to the inverse of the source packet interval T_S. The output frequency is obtained by

$$f(s) = f_n + K \cdot \bar{\delta}(s) \cdot \frac{1}{T_S}, \tag{2.17}$$

where K is a gain factor that adjusts between control speed and stability of the DPLL. Based on the NCO output the predictor estimates the virtual, global time at the arrival of

the next source packet in order to update the current clock deviation in the comparator:

$$t_r(s+1) = t_r(s) + f(s) \cdot T_S. \tag{2.18}$$

$t_r(0)$ must be initialized with the first source packet's timestamp.

2.3. Erasure Recovery

Transport layer protocols are supposed to correct packet erasures rather than single corrupted bits. Therefore, the core of the predictably reliable transport protocol is based on systematic packet-level block-erasure coding. As this coding scheme collects a block of packets from the continuous media source, it has a crucial influence on the protocol timing. Specifically, the delay introduced by a fixed-length block coding varies due to the fact that most media streams have variable source rate. This section formulates a dynamic packet-level block-erasure coding scheme that compensates for the rate variation of the media source. Since knowledge about the erasure location is essential for algebraic erasure coding, an appropriate erasure detection method makes the architecture complete.

2.3.1. Packet-level Erasure Coding

The modeling in the remainder of this thesis assumes block coding via algebraic erasure codes. The decoding success of such codes is deterministic as soon as a certain number of source or parity packets is available at the receiver. Nevertheless, an implementation based on randomized codes such as the capacity-approaching Raptor codes [142] is possible without major modifications in the protocol design. At this point, however, it is to mention that block lengths in delay-constrained media transport tend to be much shorter than those for which fountain codes benefit from the amortization of their additional overhead as well as their low complexity in coding and decoding operations.

A packet loss corresponds to the erasure of a large number of successive code symbols. Hence, block-interleaving techniques are applied in packet-level erasure coding. Specifically, virtual block interleaving ensures that the corresponding interleaving delay appears only at the receiving host since the source packets are sent to the network as soon as they are generated.

Optimal Erasure Codes

Algebraic codes are beneficial for block-erasure coding as they recover symbols of multiple bits regardless of the actual number of bit errors. The symbols are defined as elements of a Galois field [96]. Galois fields are finite number fields available at the order of primes or powers of primes, whereas the field order determines the number of symbols in the field. In case of network packets, bytes are represented by symbols on the $GF(2^8)$. Arithmetic operations on the field are formulated under a closure property such that addition and multiplication of field elements always result in another field element.

Algebraic block codes represent a message via fixed-size blocks of symbols. Each block is encoded and decoded independently. The encoder turns k source symbols into an n-symbol code word. Thus, the code word contains $n - k$ redundant symbols.

In its current implementation the PRRT protocol is based on maximum distance separable block-erasure codes, such as the *Vandermonde codes* [129]. These codes have two properties that are essential for packet-level erasure coding. First, a systematic form exists, where the original source symbols appear as a verbatim copy within the set of coded symbols. Second, the Vandermonde generator matrix produces a set of $n-k$ parity symbols that are a linearly independent combination of the k source symbols. Therefore, the codes are characterized as optimal erasure codes in that they recover the source symbols from any subset of k coded symbols. The Vandermonde Matrix is defined as

$$V = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ x_1 & x_2 & x_3 & \cdots & x_n \\ x_1^2 & x_2^2 & x_3^2 & \cdots & x_n^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_1^{n-1} & x_2^{n-1} & x_3^{n-1} & \cdots & x_n^{n-1} \end{bmatrix}, \tag{2.19}$$

where $x_1, \ldots, x_n$ are distinct elements of a Galois field with order n. A $k \times (n-k)$ Vandermonde matrix can be extended by a $k \times k$ identity matrix in order to obtain a $k \times n$ systematic generator matrix G_{EC} for the erasure code:

$$G_{EC} = \begin{bmatrix} 1 & 0 & \cdots & 0 & 1 & \cdots & 1 \\ 0 & 1 & \cdots & 0 & x_1 & \cdots & x_{n-k} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & x_1^{k-1} & \cdots & x_{n-k}^{k-1} \end{bmatrix}. \tag{2.20}$$

Let $\vec{m}$ be a message vector of k symbols. The multiplication of $\vec{m}$ with the generator matrix G_{EC} yields the code word $\vec{c}$ of length $n > k$:

$$\vec{c} = \vec{m} \cdot G_{EC}. \tag{2.21}$$

Let $\hat{\vec{c}}$ be the received code word with k symbols arbitrarily chosen from $\vec{c}$. Further, let $\hat{G}_{EC}$ be a $k \times k$ sub matrix of G_{EC} containing the columns corresponding to the chosen symbols in $\hat{\vec{c}}$. As a result of the Vandermonde property, each $k \times k$ sub matrix of G_{EC} is invertible. Consequently, the original message is recovered by simply applying the inverse of $\hat{G}_{EC}$:

$$\vec{m} = \hat{\vec{c}} \cdot \hat{G}_{EC}^{-1}. \tag{2.22}$$

In case $\hat{\vec{c}}$ contains less than k symbols, $\vec{m}$ cannot be recovered. However, due to the systematic encoding the receiver can benefit from those source symbols that have been received.

Virtual Interleaving

Transport protocols repair complete transmission segments such that the algebraic coding should be performed on packet level. As a desired property of packet-level erasure coding, the coder generates $n-k$ repair packets out of k source packets, whereas each repair packet is able to replace one of the lost source packets in the decoding process. This property results from the combination of algebraic block-erasure codes with a *virtual interleaving* technique [52, 92]. Interleaving is a common technique in channel coding in order to de-correlate continuous error bursts appearing on the communication channel.

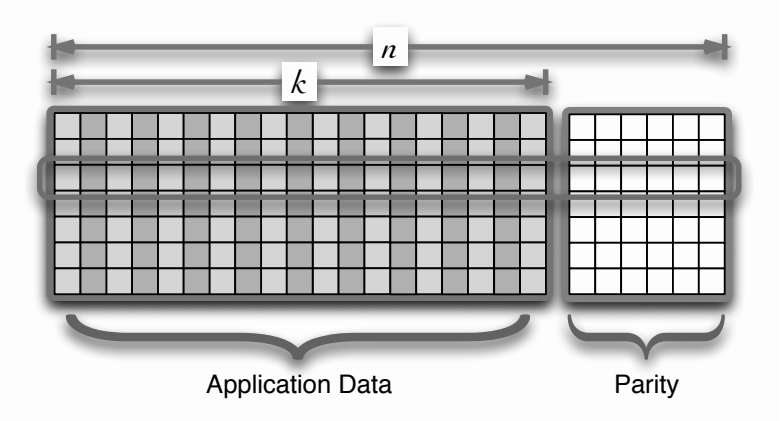


Figure 2.9.: Virtual interleaver.

The virtual interleaving is performed as follows (Figure 2.9): First, source packets are inserted into the interleaver column-wise. As soon as a complete block of packets is available, the code is applied row-wise such that each code word contains exactly one byte from each source packet. This configuration ensures that the receiver can recover the entire message from any k out of n coded packets. The receiver must collect k packets from the same coding block and arrange them in columns again. The decoding operation is performed row-wise where each code word contributes the recovery of one byte from each lost packet. As a result, virtual interleaving enables the correction of up to $n - k$ continuous packet erasures.

The term *virtual* refers to a specialty of this scheme since it basically inverts the idea of interleaving. The interleaved view on the packet sequence exists only temporarily at the sender at the time the coding operation is performed, whereas the non-interleaved sequence is transmitted over the communication channel. The interleaved view is restored at the receiver during the decoding, as it spreads packet erasures over several code words. The major advantage of the virtual interleaving becomes evident under the use of a systematic block-erasure code. The source packets can be transmitted at the same time as they are written into the interleaver such that the interleaving delay only appears at the receiver.

2.3.2. Dynamic Block-Erasure Coding

Each coding block is handled independently in PRRT. Therefore, it is possible to assign individual protocol parameters to each single block. In order to preserve the delivery delay for all packets in the current block, the coder must allow fine-grained dynamic parameterization.

The packet-level coder operates on a window of packets that is sliding in steps of one coding block size k over the sender's outgoing packet queue. As soon as the encoding process is activated, the coder obtains a reference to the most recent source packets of the queue and generates the parity packets. As a result of the virtual interleaving, the source packets remain unmodified. The block-erasure decoder works on the incoming packet queue of the receiver in a similar way.

This architecture reflects the protocol's loss tolerance as it makes the block-erasure

coding a complementary functionality that does not affect the flow of the source packets. It ensures continuous sending as well as continuous delivery to the receiver application, independently from the success of the block-erasure coding. However, if the decoding is not possible at the receiver before the delivery deadline of the respective block, the application experiences residual packet loss.

Obviously, the success of the block-erasure coding is very sensitive to delay. Delay is primarily caused by the collection of the k source packets as denoted by D_C in Section 2.2.2. A variable source rate results in the variation of the source packet interval or the source packet length such that D_C is variable under a constant coding block length. The packet-level coder must catch the resulting short-term variations with block-individual parametrization. D_C is being adjusted by the reliability control in order to follow the long-term variations of T_S (Chapter 5).

Variable Block Length

In order not to exceed the delivery deadline D_T for any packet within a coding block, the delay between the block's first packet entering the protocol socket and the last packet being sent must not be greater than the coding delay D_C determined for a packet interval T_S (Equation 2.7). If the packet interval increases due to decreasing source rate, the encoder must finalize the coding process in case less than k source packets are available after D_C. On the other hand, the coder must not collect more than k source packets into the same block under higher source rate. This would decrease the correction capabilities of the block-erasure code because of an increased code rate if the number of generated repair packets remains constant.

Consequently, the coding process is terminated by two events. Either the coding delay D_C is exceeded such that the block contains less than k source packets, or maximum k source packets are collected at the sender. If $t_{w,0}$ is the time where the first byte of the first packet of the current coding block enters the socket and $t_{w,k-1}$ the time where the first byte of the k^{th} packet enters the socket, block coding is being performed at time

$$min(t_{w,k-1} + T_S,\, t_{w,0} + D_C). \tag{2.23}$$

As a result of the variable block length, the correction capabilities of the block code increase for lower source rates, since the code rate is lowered in that the same number of parity packets are available for a smaller number of source packets.

Variable Packet Length

Packet-level erasure coding requires the source packets to have equal length. This precondition enables the virtual interleaving, where packets are arranged in columns and codewords are calculated in rows. In order to calculate the code over source packets of variable length, the columns of the interleaver must be filled up to equal length. The following algorithm handles source packets of variable length within PRRT.

Let L_D be the payload size of the largest packet among the k source packets. Each payload that is shorter than L_D is stuffed to this length by appending the respective number of zero bytes. The coder calculates the $n - k$ repair packets with equal payload length L_P. Note that the source packets have already been sent at their original length before the the start of the coding operation. The receiver derives the maximum source payload length $L_{D,max}$ from the length of the repair packets. For the decoding operation,

the received source packets must again be stuffed with zero bytes to the payload length $L_{D,max}$. Obviously, this algorithm requires the length information to be recovered for the lost source packets in order to remove the stuffing bytes from the restored source packets. This is possible since the coded payload length information is included into the parity packets.

2.3.3. Erasure Detection

Erasure coding relies on the assumption that the erasure position is known to the decoder. However, the detection of erasure positions in the case of asynchronous packet transmission is not trivial. Both proactive as well as reactive coding schemes require a certain amount of time in order to determine whether the transmission of a coding block is complete or whether feedback has to be sent. The loss recovery delay D_{PL} depends on the erasure detection method. Two methods are feasible for transport-layer protocols: Either a timeout decision is performed based on the knowledge of a regular packet interval or gap detection based on the packet's sequence number, which requires the reception of the subsequent packet in order to identify the erasure [145, 112].

Detection Methods

Common goal in erasure coding is to minimize the detection delay since it consumes valuable time from the time budget available for the error control [145]. In case of a continuous source packet stream with high data rate, it is feasible to apply sequence number gap detection. Therefore, each packet increments a continuous, monotonically increasing sequence number that uniquely identifies each packet or alternatively each byte. Lost packets are detected from discontinuous increments of the sequence number at the reception of a source packet. Gap detection does not perform well in presence of long bursts of erasures because the detection relies on the reception of the next packet. Therefore, long error bursts cause a detection delay of several packet intervals.

The timeout detection method sets a specific deadline for the arrival of a transmitted packet at the receiver. The deadline might be a small multiple of the packet interval. If the deadline expires, the expected packet is considered lost. This configuration makes the scheme independent from the reception of subsequent packets. However, the timeout must be chosen carefully since the timeout-based method is prone to packet jitter, which leads to the detection of false positives.

Both detection methods have to deal with the differentiation of erasures and re-ordered packets [119]. Re-ordered packets lead to false positives in the loss detection, which cause the transmission of spurious repair packets. This in turn degrades the protocol throughput. The fraction of re-ordered packets increases with higher sending rate and as a result of multi-path routing [64].

The detection of re-ordered packets has significant importance in the design of TCP. TCP's fast retransmit feature considers a segment as lost after the reception of three duplicate acknowledgments. If a segment is lost, the subsequent segments are experienced out-of-order with respect to the previously received segment. Therefore, an out-of-order segment with re-ordering depth of more than three packets is considered lost.

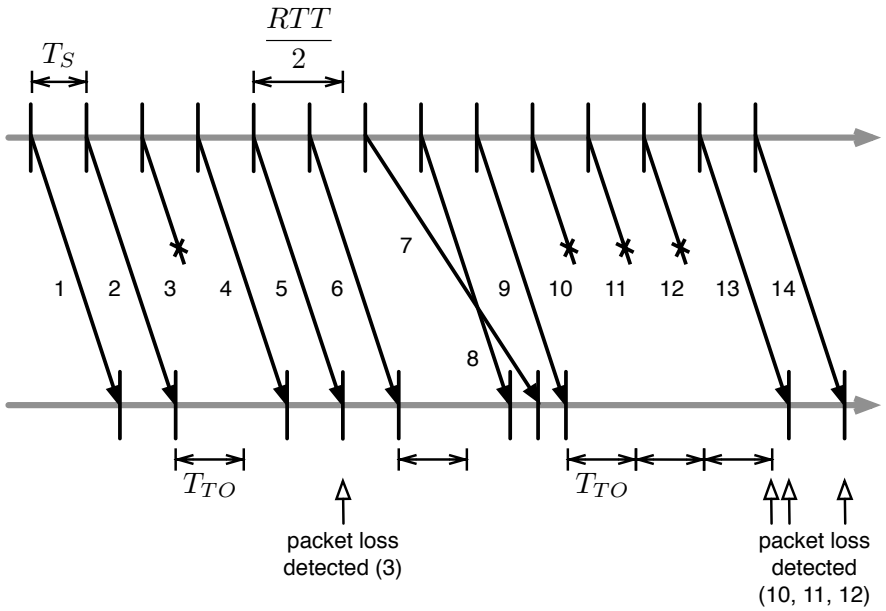


Figure 2.10.: Hybrid erasure detection [112].

Hybrid Erasure Detection

Bong Hwan [112] and Sze [145] combine erasure detection based on gap and timeout detection in order for them to compensate for each other's drawbacks. The authors introduce a voting counter for each packet. A packet is considered to be lost after it received a certain number of votes, which corresponds to a chosen re-ordering threshold ϵ. Similarly to TCP's fast retransmit, Bong Hwan chooses $\epsilon = 3$. Based on the packet interval T_S, the scheme defines a timeout of $T_{TO} = 1.5 \cdot T_S$ (Figure 2.10). The timer T_{TO} is reset after each successful packet reception or after expiration.

The algorithm expects the next packet reception to increment the latest received sequence number by one. If T_{TO} expires, the voting counter of the expected packet is incremented. For each time N_T the timer expires after successfully receiving the packet with sequence number s, the voting counter of the packets $s{+}1, \ldots, s{+}N_T$ is incremented by one. Alternatively, if an out-of-order packet is received, the difference d between the received sequence number s' and the expected sequence number $s + 1$ is calculated. In this case, the voting counter of the packets $s + 1, \ldots, s + d$ is incremented by one. As soon as the voting counter reaches the re-ordering threshold ϵ, the respective packet is considered to be lost.

PRRT implements the hybrid erasure detection scheme proposed by Bong Hwan [112]. Since the average source packet interval T_S is a known protocol parameter, the erasure detection delay is bounded by $D_{PL} = 4.5 \cdot T_S$ for a re-ordering threshold of $\epsilon = 3$.

2.4. Network Monitoring

Predictably reliable error control builds on accurate protocol timing. Therefore, the protocol's timing model must be instantly updated with measurements of the network path's round trip time. Beyond that, the current packet loss rate experienced by the

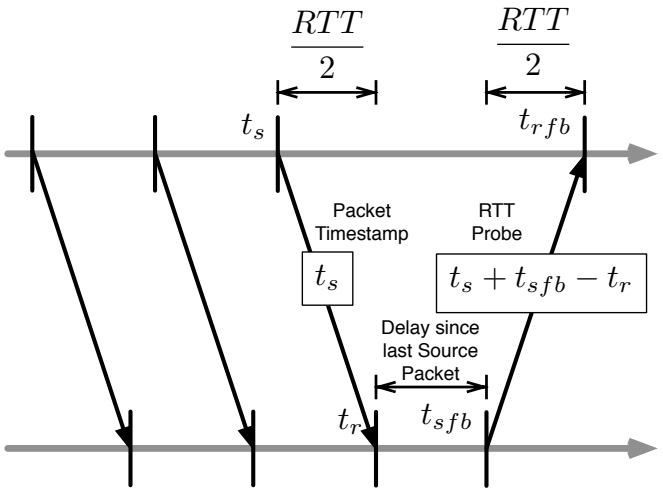


Figure 2.11.: Estimation of the round trip time.

transport protocol as well as the burstiness of the packet erasures must be known as both have significant impact on the success of the error control. The thesis refers to the joint observation of round trip delay and packet loss characteristics as the network state. The network state determines the parametrization of the protocol's timing model (Section 2.2) and the block-erasure model detailed in Chapter 3.

2.4.1. Round Trip Delay

The RTT is a crucial parameter for the scheduling of bidirectional protocol communication as well as for the receiver synchronization. PRRT estimates the RTT at the sender. The calculation is based on the **Timestamp** field of the source frame header and the **RTT probe** field in the receiver feedback. The scheme works similarly to the RTT estimation specified in NTP [106] and RTCP [135].

Estimation of the Round Trip Time

Let t_s be the local sender time at which the latest source packet leaves the outgoing packet queue (Figure 2.11). Let t_r be its arrival time at the receiver. Furthermore, define $t_{sfb} \geq t_r$ as the time where the receiver sends feedback that is received by the sender at time t_{rfb}. Based on those time stamps, the RTT can be calculated as follows:

$$\begin{aligned} RTT &= t_{rfb} - t_{sfb} + t_r - t_s \\ &= t_{rfb} - (t_s + t_{sfb} - t_r). \end{aligned} \tag{2.24}$$

The equation subtracts the delay between the reception of the source packet at t_r and the sending of the feedback packet at t_{sfb}. Hence, RTT is the sum of the network's propagation delay and the queueing delay. t_s, t_{sfb} as well as t_r are unknown at the sender. Therefore, the **RTT Probe** field of the feedback packet carries the sum $t_s + t_{sfb} - t_r$.

RTT samples are noisy due to the effect of variable queue saturation on the network path. On unmanaged Internet paths, those queue dynamics are widely caused by TCP's window control [62] as it is the prevalently applied transport protocol. This problem has

been addressed by Jacobson and Karel [79] in order to obtain good estimates for TCP's retransmission timeout. The authors proposed to catch the RTT dynamics of the network with the mean deviation, which can be applied as a conservative estimator of the standard deviation. In contrast to the standard deviation, the computational complexity of the mean deviation is low as the calculation of squares and square roots is avoided. Under normally distributed measurement noise the mean deviation over-estimates the standard deviation roughly by the factor 1.25. Jacobson's approach implements a *stochastic gradient algorithm* that is defined as follows:

$$\begin{aligned} \delta(t) &= RTT(t) - \overline{RTT}(t-1) \\ \overline{RTT}(t) &= \overline{RTT}(t-1) + \alpha \cdot \delta(t) \qquad (2.25) \\ \overline{DEV}(t) &= \overline{DEV}(t-1) + \alpha \cdot \left(|\delta(t)| - \overline{DEV}(t-1)\right). \qquad (2.26) \end{aligned}$$

Thereby, the RTT is predicted by its current average value $\overline{RTT}(t-1)$. $\delta(t)$ is the estimation error between $\overline{RTT}(t-1)$ and the most recent measurement sample $RTT(t)$. Equation 2.25 is an exponentially weighted moving average (EWMA) over the measured RTT samples with filter coefficient α. Equation 2.26 calculates the mean deviation $\overline{DEV}(t)$ likewise via EWMA over the estimation error $\delta(t)$ with gain α. It was found by Jacobson that the estimate of a retransmission timeout based on the above algorithm follows the TCP-induced variations of the RTT conservatively if four times the mean deviation $\overline{DEV}(t)$ is added to the current mean RTT while setting a filter gain of $\alpha = 0.125$ [79]. This setting allows the filters to be implemented with integer arithmetics. The predictably reliable protocol requires the RTT estimate in two qualities: The smoothed RTT estimate $\overline{RTT}(t)$ is required in the peer synchronization algorithm (Section 2.2.3) in order to compensate for the aging of the packet timestamps with $\frac{RTT}{2}$. The protocol's timing model, however, must account for the RTT's variability via reasonable delay margins. Since the request timers $D_{REQ}[c]$ of the protocol are equivalent to retransmission timers, they are calculated based on the conservative estimate $\widehat{RTT}(t)$:

$$\widehat{RTT}(t) = \overline{RTT}(t) + 4 \cdot \overline{DEV}(t).$$

Heterogeneous Propagation Delay

The above RTT estimator (Equation 2.24) measures the RTT in a centralized fashion at the sender. As long as the RTT is equal among all receivers in a multicast group, the feedback from arbitrary receivers is useful to obtain a new RTT sample. This holds true in local area networks. However, in wide are networks, the propagation delay might be heterogeneous with large deviations within the multicast group. If a receiver's RTT is larger than the current estimate at the sender, it is in disadvantage since the decoding timeout and the repair request timers are too short. In addition, under heterogeneous delay, the synchronization of the receiver clock (Section 2.2.3) relies on the exact knowledge of the individual RTT for each receiver.

One-to-many RTT estimation is not possible without sacrificing scalability. The multicast source must implement bookkeeping about each single receiver's RTT estimate and it must have dedicated communication with each receiver regularly [3]. This communication could either be implemented with a temporary point-to-point connection between multicast sender and any receiver or the sender might include the dedicated information

into the multicast stream. The latter case requires adequate signaling such that the addressed receiver can filter the information.

The estimation of each receiver's individual RTT is implemented as follows. A receiver stores the value of the RTT probe field from the header of its feedback together with the sending timestamp of the feedback. Upon reception of the feedback, the PRRT sender increments the RTT probe value by the time between feedback reception and sending of the response. The resulting value is directly sent to the respective receiver via unicast. Similarly to the RTT calculation at the sender, the receiver determines its individual RTT estimate based on the sender's response. At the same time, the receiver feedback is still useful to the sender in order to obtain a new RTT sample from the receiver group. The interval of the receiver feedback must be chosen appropriately in order not to exceed a specific session bandwidth constraint.

2.4.2. Packet Loss Rate

The packet loss rate is observed via the protocol's sequence number. Since each packet type maintains an own sequence number space, the packet loss rate can be obtained individually for source packets, repair packets and feedback packets. In order to obtain a dense sampling of the packet loss process on the forward path, the erasure positions determined for source and repair packets are jointly represented via a binary sequence at the receiver.

Loss Indicator Sequence

Each receiver maintains a binary *loss indicator sequence* $\{x_i\}_{i=1}^{N}$ with $x_i \in \{0,1\}$. The sequence represents a time series with limited history length N, where $x_i = 1$ indicates that the i_{th} packet in the series is detected as lost. The sequence is updated in first-in-first-out fashion for each newly observed packet reception or packet erasure. Under the assumption that the source packets of a multimedia stream are sent continuously, the sequence provides a sampling of the network path's packet loss process with an average period of less or equal than T_S.

The number of packet losses is obtained as the weight of the indicator sequence:

$$weight\left(\{x_i\}_{i=1}^{N}\right) = \sum_{i=1}^{N} x_i. \tag{2.27}$$

The average packet loss rate with history length N is obtained as the fraction of lost packets in the sequence:

$$PLR = \frac{weight\left(\{x_i\}_{i=1}^{N}\right)}{N}. \tag{2.28}$$

Burstiness

The loss indicator sequence contains information about the burstiness of the packet loss process. Burstiness can be described by evaluating memory in the loss process, e. g. by fitting a discrete Markov chain. Those models express the burstiness of packet erasures via the temporal correlation of subsequent packet transmissions. Section 3.2.2 gives an example for the modeling of the packet loss process via a two-state Markov chain.

Algorithm 2.1 Calculate average burst and gap length as well as average packet loss rate.

Require: X
Require: N
1: $N_e \leftarrow 0$ ▷ number of erasures
2: $N_g \leftarrow 1 - x_0$ ▷ number of gaps
3: $L_g \leftarrow N_g$ ▷ aggregate length of gaps
4: $N_b \leftarrow x_0$ ▷ number of bursts
5: $L_b \leftarrow N_b$ ▷ aggregate length of bursts
6: $G \leftarrow x_0$ ▷ in gap or in burst?
7: **for** $i = 0 \rightarrow N - 1$ **do**
8: $N_e \leftarrow N_e + x_i$
9: **if** $(G \wedge x_i)$ **then** ▷ in gap and next packet lost
10: $L_g \leftarrow L_g + 1$
11: **else if** $G \wedge \neg x_i$ **then** ▷ in gap and next packet received
12: $G \leftarrow 0$
13: $N_b \leftarrow N_b + 1$
14: $L_b \leftarrow L_b + 1$
15: **else if** $\neg G \wedge x_i$ **then** ▷ in burst and next packet lost
16: $G \leftarrow 1$
17: $N_g \leftarrow N_g + 1$
18: $L_g \leftarrow L_g + 1$
19: **else** ▷ in burst and next packet received
20: $L_b \leftarrow L_b + 1$
21: **end if**
22: **end for**
23: $PLR \leftarrow N_e/N$
24: $\bar{L}_b \leftarrow L_b/N_b$
25: $\bar{L}_g \leftarrow L_g/N_g$

In compliance with the literature [167], this thesis refers to continuous sequences of successful packet transmission as transmission bursts. Similarly, continuous sequences of packet erasure are defined as transmission gaps. Thus, a burst of length L_b is represented by an all-zero sequence $\{x_i\}_{i=1}^{L_b}$, with $x_i = 0$, whereas a gap of length L_g is represented by a sequence of ones $\{x_i\}_{i=1}^{L_g}$, with $x_i = 1$. Bursts and gaps alternate in the loss history such that the number of bursts and gaps is either equal or it differs by maximum one. The model estimator in Section 3.2.2 is based on the average run length of bursts and gaps, which are derived from the erasure history as follows.

Let the sequence $S_{01} = (0, 1)$ denote the boundary between a burst and a gap and the sequence $S_{10} = (1, 0)$ the boundary between a gap and a burst, respectively. S_{01} and S_{10} are subsequences of the loss indicator sequence $\{x_i\}_{i=1}^{N}$. Let $N_{S_{01}}$ and $N_{S_{10}}$ count the occurrences of S_{01} and S_{10} in $\{x_i\}_{i=1}^{N}$, respectively. Further, let x_1 represent the start of the sequence, where $\neg x_1$ denotes the binary complement of x_1. The number of bursts in the erasure sequence is $x_1 + N_{S_{01}}$ and the number of gaps is $\neg x_1 + N_{S_{10}}$. The average burst length $\bar{L}_b$ as well as the average gap length $\bar{L}_g$ in the indicator sequence $\{x_i\}_{i=1}^{N}$ are determined by the protocol as

$$\bar{L}_b = \frac{N - weight\left(\{x_i\}_{i=1}^{N}\right)}{x_1 + N_{S_{01}}} \text{ and } \bar{L}_g = \frac{weight\left(\{x_i\}_{i=1}^{N}\right)}{\neg x_1 + N_{S_{10}}}. \tag{2.29}$$

Algorithm 2.1 determines $N_{S_{01}}$, $N_{S_{10}}$ and $weight(\{x_i\}_{i=1}^{N})$ efficiently.

3. Packet-level Block-erasure Model

"Man tries to make for himself in the fashion that suits him best a simplified and intelligible picture of the world."
Albert Einstein

Today's Internet design represents one of the most dynamic digital communication systems. This refers not only to the variety of traffic that is delivered, but also to the heterogeneity of the underlying physical infrastructure. The characteristics of wired and wireless digital communication are fundamentally different such that transport layer protocols experience a wide range of different network state parameters. As an important fact, the packet loss process on Internet paths has memory.

Sophisticated channel models have been developed for wireless and mobile channels in order to describe the effects of multi-path reception, signal fading and doppler shift [66]. Those effects deteriorate the detection of the signal level at the receiver, which introduces bit errors into transmitted packets. The models have significantly supported system design and testing in this field of research such that the coding schemes of the physical network layer compensate for the effects of physical signal degradation with high probability, yet not exhaustively.

On packet switching networks a second source of packet loss must be considered. Those networks implement a best-effort service. Because of saturated network paths that are frequently under overload (e. g. due to TCP's loss-based congestion control), network flows experience a considerable amount of queueing losses. Due to the layered design of the Internet stack, both sources of packet errors result in packets being completely discarded at the transport layer. Depending on the network infrastructure, physically corrupted packets are already rejected at the MAC layer. The information theory refers to this behavior as an *erasure channel* [134]. Based on the work of Claude Shannon, the useful transmission rate of such erasure channels is accurately predicted [139]. This chapter modifies Shannon's well-known coding theorem in order to express the loss-tolerance of multimedia packet streams.

The general model of the erasure channel does not express the network path's memory, which is prevalently determined by the effects of mobility and massive multiplexing. As a mathematical tool, Markov chains formulate the dependence of the current packet loss characteristics on previous states of the model via conditional probabilities [104]. Yet these models require careful fitting to the observed variables so as to represent the network path with the desired accuracy. The memory of the packet erasure channel has significant impact on the performance of block-erasure coding schemes. This chapter converges into the formulation of the *block error distribution* for the mentioned block-erasure models.

3.1. Erasure Channel Model

Shannon's noisy channel coding theorem has been the initiator for extensive research in the field of information theory and channel coding. The theorem states the existence of an *arbitrarily small error probability* for a transmission at a rate R less than the *channel capacity* C, given a suitable error correcting code [43]. Therefore, the theorem assigns an explicit capacity to a communication channel, which is the fundamental limitation for reliable data transmission. It shows that principally noise on the channel does not reduce the reliability of the transmission, it rather limits the transmission rate. For rates above the channel capacity an error-free transmission is not guaranteed.

For audiovisual real-time content, totally error-free transport is not mandatory. Therefore, the general model of the packet erasure channel is extended with few modifications in order to formulate a predictable residual error. As derived in the following, the loss-tolerant model corresponds to a slightly over-utilized erasure channel.

3.1.1. Packet Erasure Channel

In compliance with the OSI layer model, each layer of the Internet stack cares for the integrity of packet data before forwarding them to the next higher layer. This holds particularly true for MAC layer and transport layer, which usually both perform an integrity check and drop corrupted transmission units. As a result, a packet appears either correctly at the subsequent layer of the stack or it appears as a packet loss. Similarly, an erasure channel either receives a symbol correctly or it receives nothing with a certain erasure probability. This model is generally applicable for IP networks since they share the property of discarding corrupted frames at lower layers or dropping frames due to queue overflow at the network layer.

Specifically, a network path is represented by the q-ary erasure channel [102]. The erasure channel models sequential symbol transmission over a communication channel, where a symbol is either received at the channel output without error or it is erased with a specific symbol erasure probability P_e (Figure 3.1). In case of the packet erasure channel, a symbol represents a network packet with payload length L_D byte from an input alphabet of size $q = 2^{(8 \cdot L_D)}$. Let the random variable I denote a symbol from the input alphabet $\{\iota_0 \dots \iota_{q-1}\}$ at the input of the erasure channel. Similarly, let Ω denote a symbol from the output alphabet $\{\omega_0 \dots \omega_{q-1}\}$ at the channel output. If the packet $\mathrm{I} = \iota_i$ with $0 \leq i \leq q-1$ is sent into the channel, packet $\Omega = \omega_i$ appears at the channel output with probability $1 - P_e$. With probability P_e the input symbol is erased, whereas the erasure position is known.

Channel Capacity

The channel capacity is a generalized upper bound for the ratio of symbols that can reliably be transferred through a noisy digital communication channel [43]. The channel capacity is derived based on a channel model that describes the translation of symbols between input and output of the communication channel. Depending on their probability distribution, input and output alphabets represent a specific average information content, which is defined to be the alphabet's entropy $H(\mathrm{I})$ and $H(\Omega)$, respectively. Communication channels are generally subject to two effects: Noise increases the entropy of the

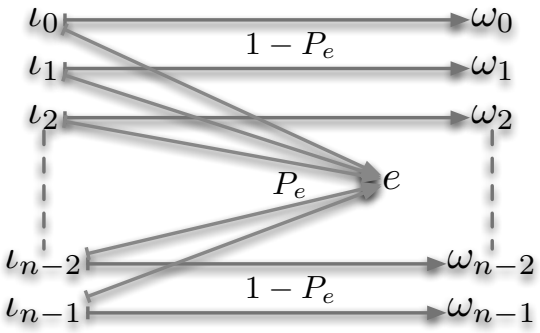


Figure 3.1.: Packet erasure channel.

channel output in case the symbol is known at the channel input. Equivocation increases the input entropy in case the channel output is known.

The channel capacity derives from the mutual information $I(\mathrm{I};\Omega)$ between input and output of the communication channel. Since input entropy $H(\mathrm{I})$ and equivocation $H(\mathrm{I}|\Omega)$ can conveniently be determined for the packet erasure channel [102], $I(\mathrm{I};\Omega)$ is formulated at the channel output as follows:

$$I(\mathrm{I};\Omega) = H(\mathrm{I}) - H(\mathrm{I}|\Omega). \tag{3.1}$$

The erasure channel does not produce noisy symbols at the output. Therefore, it is only subject to equivocation, which coincides with the occurrence of the erasure symbol e:

$$\begin{aligned} H(\mathrm{I}|\Omega) &= -\sum_{i=0}^{q} P(\Omega = \omega_i) \cdot log_q(H(\mathrm{I}|\Omega = \omega_i)) \\ &= P(\Omega = e) \cdot H(\mathrm{I}|\Omega = e) \\ &= P_\mathrm{e} \cdot H(\mathrm{I}). \end{aligned} \tag{3.2}$$

According to Shannon's theorem, the channel capacity C is the maximum mutual information between the two random variables I and Ω, which is achieved if they are chosen from an alphabet of size q with equal probability $\frac{1}{q}$. In this case the channel input has maximum entropy $H(\mathrm{I}) = q\,bits/symbol$:

$$C_{packet\_erasure} = \max_{P(\mathrm{I}=\iota_i)} (I(\mathrm{I};\Omega)) = (1 - P_\mathrm{e}) \cdot H(\mathrm{I}) = 1 - P_\mathrm{e}\,[q\,bit/symbol]. \tag{3.3}$$

Code Rate

Shannon's proof of the channel capacity is not constructive and thus it gives a purely theoretical measure. It is a major goal of the research in the field of channel coding to approach the channel capacity as closely as possible. Capacity-approaching codes have been extensively studied during the last decade. Their fundamental recipe is the support of huge code word lengths of several thousand symbols while keeping the coding complexity roughly linear via randomized coding techniques [99].

The performance of practical coding schemes is formulated via the *code rate* [43]. The code rate specifies a rate below the channel capacity at which information is transmitted if a specific code is applied to the message. For a block-erasure code the code rate is

immediately determined by the number of source symbols per code word k if the length of the code word is n:

$$R = \frac{k}{n}. \tag{3.4}$$

A block code of rate R adds $n - k$ redundancy symbols to a block of k source symbols, which decreases the channel's goodput correspondingly. The *coding overhead* or *redundancy information* is defined as

$$RI = \frac{n-k}{k}. \tag{3.5}$$

The channel capacity C is the maximum code rate of an ideal code. Error coding cannot increase the code rate R beyond C while still achieving reliable transmission. Therefore, the optimum amount of redundancy information RI_{opt} of a block-erasure code is defined as follows:

$$R \leq C_{erasure} \tag{3.6}$$

$$\Leftrightarrow \frac{k}{n} \leq 1 - P_\mathrm{e}$$

$$\Leftrightarrow \frac{n-k}{k} \geq \frac{P_\mathrm{e}}{1-P_\mathrm{e}} = RI_{opt}. \tag{3.7}$$

3.1.2. Loss-tolerant Erasure Channel

Shannon formulated the channel capacity for reliable transmission with arbitrarily small error. However, due to the nature of continuous digital media, residual packet loss is tolerable without causing a breakdown of the service. Facing the fact that such media prefer timely delivery over reliability, the residual erasure rate should be a tunable parameter of the channel model.

Channel coding corresponds to controlled rate reduction by adding redundant information in order to achieve reliability. Partially reliable coding schemes can transmit data through the erasure channel at higher rate since they add less redundancy than totally reliable codes. The loss-tolerant erasure channel model provides a measure for the channel capacity under a specific residual packet loss probability P_r [72]. An ideal predictably reliable code approaches the loss-tolerant channel capacity with residual loss probability P_r.

Cascade Erasure Channel

Let the erasure channel's packet loss probability P_e be decomposed into two probabilities P_v and P_r. P_v is a virtual packet loss probability that represents the fraction of packet erasures that is recovered by the predictably reliable protocol. P_r is the residual packet loss probability determining the fraction of symbols that is left without correction. The loss-tolerant erasure channel is modeled by a virtual cascade of two erasure channels [72, 134] with erasure probability P_v and P_r, respectively (Figure 3.2).

Reliable transmission over the first segment of the cascade is possible with maximum rate $C_v = 1 - P_v$. The second segment is offered the first segment's output rate C_v. It has a capacity of $C_r = 1 - P_r$. Therefore, the cascade channel has a joint capacity of

$$C = C_v \cdot C_r = (1 - P_v) \cdot (1 - P_r). \tag{3.8}$$

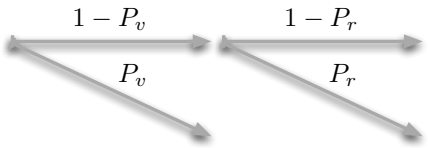


Figure 3.2.: Loss-tolerant packet erasure channel [72].

Correspondingly, the overall erasure rate P_e of the cascade channel is

$$P_e = 1 - C = 1 - (1 - P_v) \cdot (1 - P_r). \tag{3.9}$$

Solving for P_v yields the virtual erasure probability of the first segment:

$$P_v = \begin{cases} \frac{P_e - P_r}{1 - P_r} & P_e > P_r \\ 0 & else \end{cases}. \tag{3.10}$$

Channel Capacity under Residual Erasure Rate

In order to achieve a residual error probability P_r, the erasure code must only repair a loss rate of P_v. Therefore, a rate reduction of P_v is sufficient, which corresponds to the capacity C_v of the first virtual segment of the loss-tolerant model. This in turn leads to a slightly increased channel capacity C_{lt} for the loss-tolerant erasure channel in comparison to the reliable erasure channel [72]:

$$C_{lt} = C_v = 1 - P_v = \frac{1 - P_e}{1 - P_r}. \tag{3.11}$$

Due to the loss-tolerance, the minimum amount of redundancy information is reduced. For the loss-tolerant erasure channel, RI depends on the virtual erasure rate P_v because the loss-tolerant code has to be dimensioned to leave a residual loss probability of P_r:

$$RI_{lt} = \frac{P_v}{1 - P_v} = \frac{P_e - P_r}{1 - P_e}. \tag{3.12}$$

3.2. Stochastic Block-erasure Model

The transport layer protocol obtains a packet-level view of the error behavior on the network path by a continuity check of the packets' sequence numbers. Therefore, it obtains a new sample of the network state with each incoming packet. Essential parameter for the dimensioning of block-erasure codes is the packet loss probability. As the PLR is a statistical measure, it is represented by a block-erasure model that is fit to the packet loss observations. This model randomly switches between a set of states within a slotted time and assumes the transmission of fixed-length data packets. By convention, the model performs a transition right before the transmission of a packet [167].

Besides the packet erasure probability, the model conveys the burstiness of the packet erasure process if it considers the channel's memory. Under a limited number of measurements the model estimator is obtained with a specific error: As the measurement sample might insufficiently represent the population of the packet loss process, the model potentially deviates from the true packet loss probability of the error process. The estimator's accuracy is therefore being addressed in the following sections.

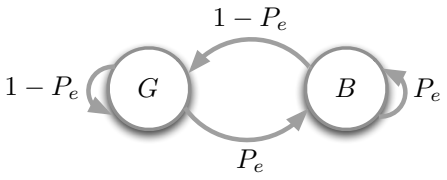


Figure 3.3.: Bernoulli model.

3.2.1. Independent Packet Loss

On a memoryless packet erasure channel the packet losses are assumed to be independently and identically distributed (i. i. d.) over time. Suchlike network paths are modeled by a Bernoulli process with packet erasure probability P_e.

Bernoulli Model

The Bernoulli model (Figure 3.3) defines a binary random variable $X \in \{0, 1\}$, where $X = 1$ represents packet loss and $X = 0$ successful reception. Thus, it is entirely characterized by the packet loss probability P_e, whereas the complement $1 - P_e$ is the probability of successful packet reception. The probability mass function of the Bernoulli model is

$$Pr(X = x_i; P_e) = P_e^{x_i}(1 - P_e)^{1-x_i}. \tag{3.13}$$

The model describes a two-state transition system without memory (Figure 3.3). Let a packet loss event be represented by a *bad* state B and successful reception by a *good* state G, respectively. As subsequent transmission events are statistically independent, the system enters state B with probability P_e independently from the previous state. Similarly, it enters state G with $1 - P_e$ from any previous state. If the model resides in state G, it generates a transmission burst. While staying in B, it represents a gap of several packet erasures. Let L_b and L_g be random variables for the burst and gap length in a time series of packet transmissions (Section 2.4.2). The Bernoulli model generates L_b and L_g with i. i. d. geometric distribution and the following probability mass functions

$$\begin{aligned} Pr(L_b = i; P_e) &= (1 - P_e)^{(i-1)} P_e \\ Pr(L_g = j; P_e) &= P_e^{(j-1)}(1 - P_e). \end{aligned} \tag{3.14}$$

A burst is terminated with a transition into state B after $L_b - 1$ self-transitions into state G and vice versa for a transmission gap.

Model Estimator

The Bernoulli model is fit to a concrete network observation via maximum-likelihood estimation (MLE) [50]. The goal of this method is to find model parameters that produce a packet loss distribution that generates the observed packet loss distribution with maximum probability. In the following a maximum likelihood estimator for the Bernoulli model's single parameter, the packet loss probability P_e, is being derived.

Let X be a random variable with $X \in \{0, 1\}$ that follows a Bernoulli distribution with estimated parameter $\hat{P}_e$. Let $(x_1, x_2, ..., x_N)$ represent N observations from a memoryless

erasure channel that are statistically independent, where $x_i = 1$ denotes a packet erasure. Each observation x_i corresponds to a potential instance of the Bernoulli model's probability mass function with success probability P_e. MLE specifies an estimator function that describes the joint distribution of the N observations. Under the assumption that the different samples are statistically independent, the estimator function L_{P_e} is obtained as a product over N instances of the probability mass function of X:

$$L_{P_e} = \prod_{i=1}^{N} Pr(X = x_i; \hat{P}_e) = \hat{P}_e^{\sum x_i} \cdot (1 - \hat{P}_e)^{N - \sum x_i}. \tag{3.15}$$

L_{P_e} must be maximized under the given observation. Therefore, the first derivative of L_{P_e} with respect to P_e is calculated:

$$\frac{\partial L_{\hat{P}_e}}{\partial \hat{P}_e} = \left(\sum x_i\right) \cdot \hat{P}_e^{\sum x_i - 1} \cdot (1 - \hat{P}_e)^{N - \sum x_i} - \hat{P}_e^{\sum x_i} \left(N - \sum x_i\right) \cdot (1 - \hat{P}_e)^{N - \sum x_i - 1}.$$

Setting the derivative to zero while assuming $0 < \hat{P}_e < 1$, yields

$$\begin{aligned} & \hat{P}_e^{\sum x_i} \cdot (1 - \hat{P}_e)^{N - \sum x_i} \cdot \left[\frac{\sum x_i}{\hat{P}_e} - \frac{N - \sum x_i}{1 - \hat{P}_e} \right] = 0 \qquad (3.16) \\ \Leftrightarrow \quad & \frac{\sum x_i}{\hat{P}_e} - \frac{N - \sum x_i}{1 - \hat{P}_e} = 0 \\ \Leftrightarrow \quad & (1 - \hat{P}_e) \sum x_i + \hat{P}_e \sum x_i = N \hat{P}_e \\ \Leftrightarrow \quad & \frac{\sum x_i}{N} = \hat{P}_e. \end{aligned}$$

Consequently, the maximum likelihood estimator for $\hat{P}_e$ is defined as the sample mean of X:

$$\hat{P}_e = \bar{X} = \frac{1}{N} \sum_{i=1}^{N} x_i. \tag{3.17}$$

Accuracy of the Bernoulli Estimator

A statistical model represents a population of values characterized by specific statistical properties such as a certain mean and a certain standard deviation. However, a random sample taken from the population does not necessarily reproduce the true mean value. The Bernoulli model assumes the N measurements of the estimator to be uncorrelated such that the sample mean is normally distributed around the population mean P_e [16] with a standard error of

$$SE = \frac{s}{\sqrt{N}} = \sqrt{\frac{\hat{P}_e \left(1 - \hat{P}_e\right)}{N}}, \tag{3.18}$$

where $s = \sqrt{\hat{P}_e(1 - \hat{P}_e)}$ is the estimate of the population standard deviation. In order to achieve a confidence level P_C for the estimate of the mean, a confidence interval of ν multiples of the standard error SE is defined around $\hat{P}_e$ with

$$\nu = \sqrt{2} \cdot erf^{-1}(P_C). \tag{3.19}$$

$erf^{-1}(x)$ is the inverse error function. For instance, it yields $\nu \approx 2.58$ for a 99 % confidence level. The confidence interval of the Bernoulli estimator is formulated as (compare [166])

$$\hat{P}_e - \nu \cdot \sqrt{\frac{\hat{P}_e\left(1-\hat{P}_e\right)}{N}} \leq P_e \leq \hat{P}_e + \nu \cdot \sqrt{\frac{\hat{P}_e\left(1-\hat{P}_e\right)}{N}}. \tag{3.20}$$

The relationship between the estimator's accuracy and the number of samples N required to fit the model so as to approach the population mean with high probability is of particular interest for the protocol implementation in order to dimension the sample history of the network observation. Let a_{P_e} be the accuracy of the estimator with $(1-a_{P_e}) \cdot \hat{P}_e \leq P_e \leq (1+a_{P_e}) \cdot \hat{P}_e$ and $a_{P_e} < 1$. This leads to

$$a_{P_e}^2 \leq \left(\frac{P_e - \hat{P}_e}{\hat{P}_e}\right)^2. \tag{3.21}$$

Using upper and lower boundary of the confidence interval (Equation 3.20) yields

$$\begin{aligned} a_{P_e}^2 &\leq \left(\frac{1}{\hat{P}_e}\right)^2 \cdot \left(\pm\nu \cdot \sqrt{\frac{\hat{P}_e\left(1-\hat{P}_e\right)}{N}}\right)^2 \\ \Rightarrow a_{P_e} &\leq \left|\nu \cdot \sqrt{\left(\frac{1}{\hat{P}_e} - 1\right) \cdot \frac{1}{N}}\right|. \end{aligned} \tag{3.22}$$

In order to achieve a specific accuracy of $\pm a_{P_e}$ with confidence level P_C,

$$N = \left(\frac{1}{\hat{P}_e} - 1\right) \cdot \left(\frac{\nu}{a_{P_e}}\right)^2 \tag{3.23}$$

independent observations are required. N depends on the estimate of the erasure rate $\hat{P}_e$ itself such that the problem is recursive. However, starting from an initial, sufficiently large sample, $\hat{P}_e$ can be estimated iteratively by refining N within each step. Figure 3.4 shows the required sample size for the estimation of the packet loss rate with different accuracy. For instance, the estimation of a packet loss rate of 1 % requires roughly 16500 independent measurement samples in order to reach an accuracy of ± 0.2 with 99 % confidence level.

3.2.2. Correlated Packet Loss

Temporal correlation characterizes the memory in the packet erasure process. Usually, correlation is significant up to a certain maximum distance between any two measurement samples, i. e. the measurement time series has a self-similarity up to a certain lag. Beyond this lag, samples of the packet loss process are considered to be statistically independent. Markov chain models are common tools to express the correlation in a statistical process [104]. In particular, non-hidden Markov chains of order h directly express the temporal dependency of the current state of the model on the h previous states.

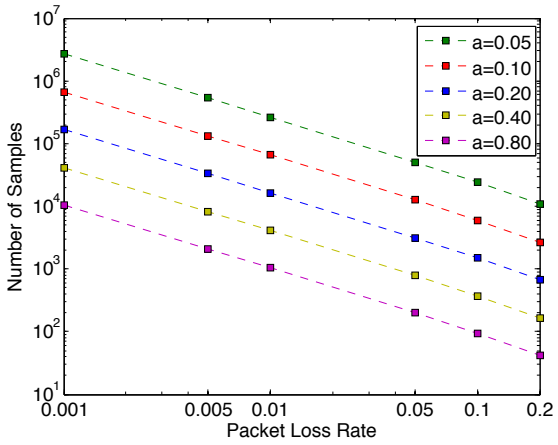


Figure 3.4.: Required sample size in order to estimate the packet loss rate with a desired accuracy $\pm a$ for a 99 % confidence interval.

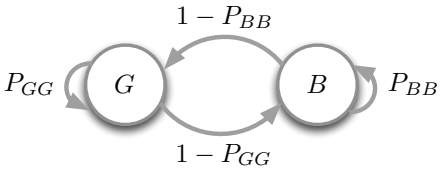


Figure 3.5.: Simplified Gilbert-Elliott model.

Simplified Gilbert-Elliott Model

The Gilbert-Elliott (GE) model [65, 51] is widely applied in theoretical work to express memory in the packet erasure process [10, 147, 169, 149, 166]. The model comprises a two-state Markov chain that models a state of high error rate with a *bad* state B and low error rate with a *good* state G. Specifically, the simplified GE model (Figure 3.5) assigns an erasure probability of 1 to state B and 0 to state G. The model is completely described by the transition matrix P_t. The matrix is determined by two transition probabilities P_{BB} and P_{GG} and their complements, where P_{BB} is the self-transition probability for state B and P_{GG} for state G, respectively. A state transition of the GE model depends only on the current state. Hence, P_{BB} and P_{GG} are conditional probabilities:

$$P_t = \begin{bmatrix} P_{GG} & 1 - P_{GG} \\ 1 - P_{BB} & P_{BB} \end{bmatrix}. \tag{3.24}$$

The steady state probabilities P_G and P_B measure the fraction of time in which the model resides in states G and B, respectively:

$$P_G = \frac{1 - P_{BB}}{2 - P_{BB} - P_{GG}}, \; P_B = \frac{1 - P_{GG}}{2 - P_{BB} - P_{GG}}. \tag{3.25}$$

Since the simplified model sets $P_{e,B} = 1$ and $P_{e,G} = 0$, the network's erasure probability is directly obtained from the steady state probability P_B:

$$P_e = P_G \cdot P_{e,G} + P_B \cdot P_{e,B} = P_B. \tag{3.26}$$

Let S be a random variable that denotes the state of the GE model as follows

$$S = \begin{cases} 1 & \text{if model is in state } B \\ 0 & \text{if model is in state } G \end{cases}. \tag{3.27}$$

In order to measure the burstiness of the model, ρ is defined as the correlation coefficient of two consecutive states of the model [167, 94]:

$$\begin{aligned} \rho &= \frac{E\left[(S_i - P_e)\,(S_{i+1} - P_e)\right]}{\sigma^2} \\ &= \frac{E[S_i S_{i+1}] - P_e E[S_i] - P_e E[S_{i+1}] + P_e^2}{P_e\,(1 - P_e)} \\ &= \frac{P_e\,(P_{BB} - P_e)}{P_e\,(1 - P_e)}, \end{aligned} \tag{3.28}$$

where S_i and S_{i+1} denote subsequent states of the model. Substituting P_e with Equations 3.25 and 3.26 yields

$$\rho = P_{BB} + P_{GG} - 1. \tag{3.29}$$

After coordinate transformation, the model is expressed via the erasure rate P_e and the correlation coefficient ρ:

$$\begin{aligned} P_{BB} &= \rho + P_e \cdot (1 - \rho) \\ P_{GG} &= 1 - P_e \cdot (1 - \rho). \end{aligned} \tag{3.30}$$

The sojourn times of the simplified GE model correspond to the average burst length as well as the average gap length, which are directly retrievable by measurement. Let L_b and L_g be random variables for the observed burst and gap length, respectively. L_b and L_g follow an i. i. d. geometric distribution with the probability mass functions

$$\begin{aligned} Pr(L_b = i; P_{GG}) &= P_{GG}^{(i-1)}(1 - P_{GG}), \\ Pr(L_g = j; P_{BB}) &= P_{BB}^{(j-1)}(1 - P_{BB}) \end{aligned} \tag{3.31}$$

and mean

$$\begin{aligned} E(L_b) &= \sum_{i=1}^{\infty} i(1 - P_{GG})P_{GG}^{(i-1)} = \frac{1}{1 - P_{GG}}, \\ E(L_g) &= \sum_{j=1}^{\infty} j(1 - P_{BB})P_{BB}^{(j-1)} = \frac{1}{1 - P_{BB}}. \end{aligned} \tag{3.32}$$

Model Estimator

As for the Bernoulli model, the GE estimator is conveniently derived via maximum-likelihood estimation [146]. The maximum-likelihood estimator of the GE self-transition probabilities is based on the observation of the burst and gap lengths. In the following,

the estimator for the self-transition probability $\hat{P}_{GG}$ into state G is being derived from measurements of the burst length L_b. Due to the symmetry of the GE model, the proof is equivalent for the estimator of $\hat{P}_{BB}$ via sampling of the gap length L_g. Let $(l_1,l_2,...,l_N)$ be N statistically independent observations for L_b with $L_b \in \mathbb{N}$. Each observation l_i corresponds to a potential instance of the geometric probability mass function of the burst length $Pr(L_b = i; \hat{P}_{GG})$. The estimator function L_l is formulated as the joint probability of the independent observations:

$$\begin{aligned} L_{\hat{P}_{GG}} &= \prod_{i=1}^{N} Pr(L_b = i; \hat{P}_{GG}) = \prod_{i=1}^{N} \hat{P}_{GG}^{(l_i-1)} \cdot (1 - \hat{P}_{GG}) \\ &= \hat{P}_{GG}^{(\sum_{i=1}^{N} l_i - N)} \cdot (1 - \hat{P}_{GG})^N. \end{aligned} \tag{3.33}$$

The maximization of $L_{\hat{P}_{GG}}$, corresponds to the maximization of $ln(L_{\hat{P}_{GG}})$ since the logarithm is a continuous, strictly increasing function. However, the logarithm significantly simplifies the analysis of the subsequent steps:

$$ln(L_{\hat{P}_{GG}}) = N \cdot ln(1 - \hat{P}_{GG}) + \left(\sum_{i=1}^{N} l_i - N \right) \cdot ln(\hat{P}_{GG}). \tag{3.34}$$

Finally, the maximum of $ln(L_{\hat{P}_{GG}})$ is determined:

$$\begin{aligned} \frac{\partial ln(L_{\hat{P}_{GG}})}{\partial \hat{P}_{GG}} &= \frac{\left(\sum_{i=1}^{N} l_i - N\right)}{\hat{P}_{GG}} - \frac{N}{1 - \hat{P}_{GG}} = 0 \\ &\Leftrightarrow \sum_{i=1}^{N} l_i - N = \frac{\hat{P}_{GG} \cdot N}{1 - \hat{P}_{GG}} \\ &\Leftrightarrow \frac{1}{N} \cdot \sum_{i=1}^{N} l_i = \frac{1}{1 - \hat{P}_{GG}} \\ &\Leftrightarrow \hat{P}_{GG} = 1 - \frac{N}{\sum_{i=1}^{N} l_i}. \end{aligned}$$

The maximum likelihood estimator for the self-transition probability $\hat{P}_{GG}$ depends on the average over the observed samples of L_b:

$$\hat{P}_{GG} = 1 - \frac{1}{\bar{L}_b} = 1 - \frac{N}{\sum_{i=1}^{N} l_i}. \tag{3.35}$$

$\sum_{i=1}^{N} l_i$ is interpreted as the total number of transitions into the state the self-transition probability is estimated for. N reflects the number of transitions coming from the opposite state. Therefore, the estimator of the self-transition probabilities of the simplified GE model is a Bernoulli experiment modeling the average number of self-transitions among the total number of transitions into the considered state.

Accuracy of the GE Estimator

Similarly to the Bernoulli estimator, the GE estimators are derived from the protocol's packet loss indicator sequence (Section 2.4.2). Let N be the length of the loss indicator

sequence. As both $\hat{P}_{BB}$ and $\hat{P}_{GG}$ are considered to be the parameters of Bernoulli processes describing the probability of self-transitions into state G and B, respectively, their mean estimates are normally distributed around the population's parameters P_{BB} and P_{GG} with the standard errors

$$SE_{P_{BB}} = \sqrt{\frac{\hat{P}_{BB}\left(1-\hat{P}_{BB}\right)}{N\cdot\hat{P}_e}} \text{ and } SE_{P_{GG}} = \sqrt{\frac{\hat{P}_{GG}\left(1-\hat{P}_{GG}\right)}{N\cdot\left(1-\hat{P}_e\right)}}. \tag{3.36}$$

Note that the number of transitions into state B can be expressed by $N\cdot\hat{P}_e$ in case of the simplified GE model. Equivalently, there are $N\cdot(1-\hat{P}_e)$ transitions into state G. A confidence interval with confidence level P_C for both estimators is therefore formulated as (compare [166])

$$\hat{P}_{BB} - \nu\cdot\sqrt{\frac{\hat{P}_{BB}\left(1-\hat{P}_{BB}\right)}{N\cdot\hat{P}_e}} \leq P_{BB} \leq \hat{P}_{BB} + \nu\cdot\sqrt{\frac{\hat{P}_{BB}\left(1-\hat{P}_{BB}\right)}{N\cdot\hat{P}_e}} \tag{3.37}$$

and

$$\hat{P}_{GG} - \nu\cdot\sqrt{\frac{\hat{P}_{GG}\left(1-\hat{P}_{GG}\right)}{N\cdot\left(1-\hat{P}_e\right)}} \leq P_{GG} \leq \hat{P}_{GG} + \nu\cdot\sqrt{\frac{\hat{P}_{GG}\left(1-\hat{P}_{GG}\right)}{N\cdot\left(1-\hat{P}_e\right)}},$$

where ν is obtained via Equation 3.19. Let $a_{P_{GG}}$ and $a_{P_{BB}}$ be the accuracies of both estimators (Section 3.2.1):

$$a_{P_{GG}} \leq \left|\nu\cdot\sqrt{\frac{(1-\hat{P}_{GG})}{\hat{P}_{GG}\cdot(1-\hat{P}_e)}\cdot\frac{1}{N}}\right| \tag{3.38}$$

$$a_{P_{BB}} \leq \left|\nu\cdot\sqrt{\frac{(1-\hat{P}_{BB})}{\hat{P}_{BB}\cdot\hat{P}_e}\cdot\frac{1}{N}}\right|.$$

Consequently, a history of

$$N_{P_{GG}} = \frac{(1-\hat{P}_{GG})}{\hat{P}_{GG}\cdot(1-\hat{P}_e)}\cdot\left(\frac{\nu}{a_{P_{GG}}}\right)^2 \tag{3.39}$$

and

$$N_{P_{BB}} = \frac{(1-\hat{P}_{BB})}{\hat{P}_{BB}\cdot\hat{P}_e}\cdot\left(\frac{\nu}{a_{P_{BB}}}\right)^2$$

samples must be maintained at the receiver in order to achieve confidence level P_C with accuracy $\pm a_{P_{GG}}$ and $\pm a_{P_{BB}}$, respectively. However, since bursts and gaps alternate, their number differs by maximum one such that the network observation (i. e. the loss indicator sequence) should be dimensioned in order to satisfy $N = max(N_{P_{GG}}, N_{P_{BB}})$. Figure 3.6 shows that the required sample size increases by roughly one order of magnitude in order to obtain the GE estimator with the same accuracy as the Bernoulli estimator (compare Figure 3.4).

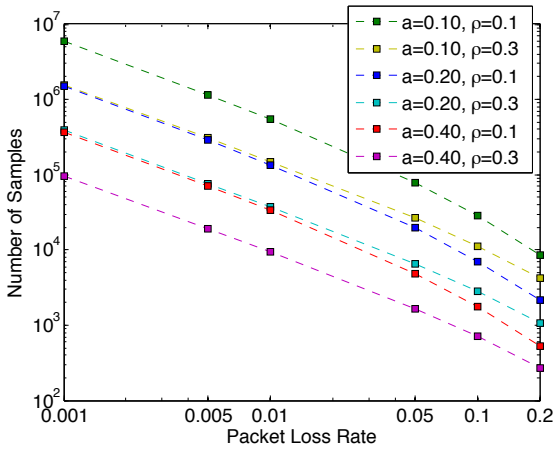


Figure 3.6.: Required sample size in order to estimate the GE model parameters with a desired accuracy of $\pm a$ for a 99 % confidence interval.

3.2.3. Queueing Loss

Physical packet corruption has negligible contribution to the erasure rate in wide area network paths[1] as long as they do not include wireless or mobile segments. Packet loss is usually the result of saturated queues during flow multiplexing at Internet routers. Those queues usually implement first-in-first-out and drop-tail policies for best-effort traffic [41]. As a result, packet losses are particularly bursty for continuous packet flows if they experience a saturated queue since the state of saturation persists until the transport protocols react appropriately to drain the queue [38]. Time series analysis on Internet packet loss traces has shown that Markov models with two or more states can express the burstiness of queueing loss [166, 22]. However, in case the related system parameters such as queue length and saturation of the intermediate devices are known or can be estimated accurately, queueing models provide a more realistic simulation of the packet loss behavior. Queueing models on finite first-in-first-out queues formulate a specific blocking probability reflecting the fraction of time in which a flow finds the queue in saturated state such that the packets are rejected.

Queueing Model

The single multiplexer model [19] simulates the effects of several packet flows sharing a single network bottleneck. In most cases, the flows are assumed to induce packets with i. i. d. exponentially distributed inter-arrival times. The service times reflecting the throughput of the bottleneck are assumed to follow an exponential distribution as well. This corresponds to Poisson distributions with parameters λ and μ [86], respectively, for the arrival and service rates. These assumptions simplify the model significantly since the queue dynamics can be expressed via a birth-death Markov process (Figure 3.7), where the queue level increments with the arrival rate λ and decrements with the service rate μ [17]. The Markov chain represents each queue level with a dedicated state $j = 0, 1, 2, ...$ If the buffer space at the multiplexer is limited, the queue may expand to a maximum

[1] Bit error rates of 10^{-14} have been measured on long-distance fiber [38], which lead to packet erasure rates around 10^{-10} at 1400 *byte* packet size.

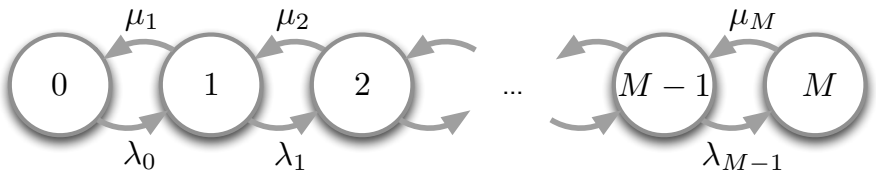


Figure 3.7.: Birth-death process.

length M, which results in $M+1$ different states $0 \leq j \leq M$ for the Markov model. As a result of the limited queue length, packets are lost when they arrive at the queue while it is in state M.

Whereas the Markovian queueing model reflects bulk traffic in large-scale multiplexing systems very well, it might be too conservative for some scenarios. Therefore, more sophisticated models are built upon specific probability distributions for λ and μ, while their analysis turns out to be significantly more complex [17]. Since the complexity of various queueing models cannot be covered within the scope of this thesis, it limits the focus on the above queueing model as an exemplary instance of the block-erasure model. If required, it might be replaced by a more sophisticated model as the block-erasure model is conveniently exchangeable in the developed protocol architecture.

Saturation and Blocking Probability

Let N be the number of flows multiplexed at the bottleneck. The i^{th} flow, $0 < i \leq N$ offers the rate λ_i, whereas the multiplexer processes the packets at a service rate μ. The superposition of N Poisson processes with parameters $\lambda_1, \ldots, \lambda_N$ results in another Poisson process with parameter $\lambda_S = \sum_{i=1}^{N} \lambda_i$ [17]. This corresponds to an aggregate packet flow with arrival rate λ_S. Both the arrival process and the service process determine the global balance of the system: If P_{j-1} and P_j are the steady state probabilities of two subsequent queue levels $j-1$ and j, respectively, the following relationship holds:

$$\lambda_S \cdot P_{j-1} = \mu \cdot P_j \Leftrightarrow P_j = \frac{\lambda_S}{\mu} \cdot P_{j-1}. \tag{3.40}$$

Correspondingly, the saturation or average load θ of the multiplexer is defined as

$$\theta = \frac{\lambda_S}{\mu}. \tag{3.41}$$

A necessary condition for the queueing system to be stable is $\theta < 1$. The queue has level $j \geq 0$ with probability

$$P_j = \theta^j \cdot P_0. \tag{3.42}$$

where P_0 is the probability of the queue being empty.

For the remainder of the thesis, just systems with a limited buffer space of M positions are of interest. As such systems must reside in one of the $M+1$ different queue states at any time, the probability of an empty queue is obtained as

$$\sum_{j=0}^{M} P_j = 1 \Leftrightarrow \sum_{j=0}^{M} \theta^j P_0 = 1 \Rightarrow P_0 = \frac{1-\theta}{1-\theta^{M+1}}. \tag{3.43}$$

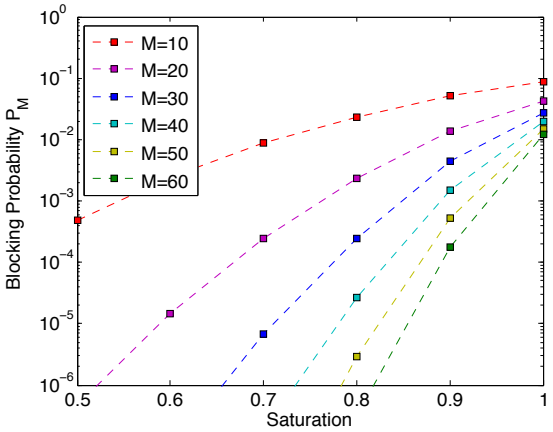


Figure 3.8.: Blocking probability for different buffer sizes M under different saturation θ.

Consequently, given the system's saturation coefficient θ, the *blocking probability*, which is the probability for a packet to observe the queue in saturated state, is formulated as follows:

$$P_M = \theta^M \cdot P_0 = \frac{(1-\theta)\theta^M}{1-\theta^{M+1}}. \tag{3.44}$$

Figure 3.8 shows the blocking probability in dependence of the queue saturation for different buffer sizes. In general, a saturated multiplexer ($\theta = 1$) produces a few percent of packet loss, whereas the impact of the buffer size is small. For highly utilized systems ($\theta \approx 0.9$), however, the level of reliability grows exponentially with the buffer size.

3.3. Block Error Distribution

The block error distribution $P_m(e, m)$ is defined as the probability of losing e out of m packets with respect to the block-erasure model [167]. The distribution can therefore express the impact of the memory in the packet loss process, which particularly affects short block-erasure codes. In the architecture of predictable reliability $P_m(e, m)$ is the single interface between the block-erasure model and the protocol's parameter adaptation. Therefore, the model can conveniently be replaced without modifying the remaining protocol architecture. In case of the memoryless Bernoulli model, $P_m(e, m)$ is immediately obtained from the probability mass function of the binomial distribution

$$P_m\left(e, m\right) = \binom{m}{e} P_e^e \left(1 - P_e\right)^{m-e}. \tag{3.45}$$

For the Gilbert-Elliott model as well as the queueing model the history of previous states must be considered, such that the derivation of $P_m(e, m)$ is considerably more complex.

3.3.1. Gilbert-Elliott Sequence Analysis

For a sequence length m the Gilbert-Elliott model generates 2^m different traces of state transitions. Each trace passes the bad state for a certain number of times, leading to

the corresponding number of packet erasures. A convenient solution to obtain the block error distribution is a recursion that walks through all possible state transitions while considering the respective transition probabilities. However, the recursion causes high computational complexity and the block error distribution does not explicitly include information on the exact sequence of state transitions. As several traces of state transitions appear with equal probability, the block error distribution can be generated more efficiently via a combinatorial approach. Based on the combinatorial formulation, a closed form equation for $P_m(e, m)$ is being derived.

Recursion Analysis

The block-error distribution $P_m(e, m)$ can be calculated by a recursion formulated over the transition probabilities P_{BB} and P_{GG} [167]. The recursion formula models the probability of e erasures starting from the end of a sequence of length m, where the sequence either terminates in state G or in state B. Depending on the final state, the recursion formula evolves in two branches $P_G(e, m)$ and $P_B(e, m)$.

Under the assumption that the simplified GE model (Section 3.2.2) is applied, erasures are only produced in state B. Consequently, the recursion decrements the number of erasures e only after transition into state B. The overall recursion is formulated as follows:

$$\begin{aligned} P_m(e, m) &= P_{m,G}(e, m) + P_{m,B}(e, m) \\ P_{m,G}(e, m) &= P_{m,G}(e, m-1)P_{GG} + P_{m,B}(e, m-1)(1 - P_{BB}) \\ P_{m,B}(e, m) &= P_{m,B}(e-1, m-1)P_{BB} + P_{m,G}(e-1, m-1)(1 - P_{GG}). \end{aligned} \tag{3.46}$$

The recursion is initialized with the following conditions:

$$\begin{aligned} P_{m,G}(e, 0) &= P_{m,B}(e, 0) = 0,\ e \neq 0 \\ P_{m,G}(0, 0) &= P_G \\ P_{m,B}(0, 0) &= P_B. \end{aligned}$$

P_G and P_B are the steady state probabilities of the simplified GE model. The calculation of $P_m(e, m)$ requires up to $m \cdot (m + 1)$ different evaluations of $P_{m,G}$ and $P_{m,B}$, respectively, where each evaluation includes two multiplications and one addition. In case the implementation of the recursive algorithm stores intermediate results for $P_{m,G}$ and $P_{m,B}$, an overall quadratic complexity with $4 \cdot m \cdot (m + 1)$ multiplications and $2 \cdot m \cdot (m + 1)$ additions is achievable with a space requirement of $O(2m^2)$.

Figure 3.9 shows the block-error distribution for two different sequence lengths m under various combinations of the packet loss probability P_e and the correlation coefficient ρ. Whereas m and P_e determine the distribution's mean, ρ influences the dispersion of the probability density: The greater the correlation, the higher the probability for erasure lengths less and greater than the mean value. For the extreme case of $\rho = 1$ the density converges to two Dirac delta pulses at $e = 0$ and $e = m$, weighted by P_G and P_B, respectively.

Combinatorial Analysis

The quadratic complexity of the recursion analysis renders online calculation of the block-erasure distribution difficult. A closed form solution for the probability mass function

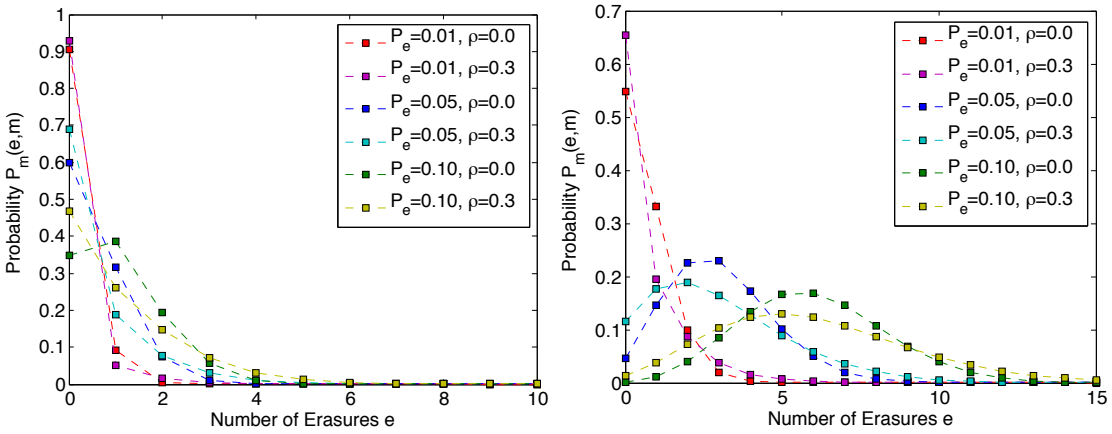


Figure 3.9.: Block-error distribution for sequence length $m = 10$ (left) and $m = 60$ (right, truncated to $e = 15$) with and without temporal correlation.

$P_m(e, m)$ on a simplified GE path is computed as follows [167]. A transmission sequence of m packets is considered, in which e packets are lost and $m - e$ packets are received successfully. The e erasures may occur at arbitrary positions within this sequence. The number of different patterns of e erasures and $m - e$ successfully received packets is $\binom{m}{e}$. Under binomially distributed erasures (Equation 3.45), each such pattern occurs with an equal probability $P_e^e(1 - P_e)^{m-e}$. However, in case of a simplified GE model, the patterns have different probabilities, determined by the number of gaps g, the number of bursts b and the number of erasures e. This thesis refers to gaps as continuous sequences of packet erasures. Similarly, continuous sequences of successful packet transmission are considered to be bursts.

The analysis by Yee and Weldon [167] differentiates between four different cases to derive the probability of occurrence $Q(g, b, e, m)$ of each pattern and their number of occurrences $N(g, b, e, m)$. Each of the four cases corresponds to a different combination of g, b, e and m. Note that bursts and gaps alternate, whereas a sequence either starts with a burst or a gap. Therefore, the four cases are given as follows:

1. The sequence starts with a gap and ends with a burst, which implies $g = b$. The model performs g transitions from state B to G and $g - 1$ transitions from state G to B. It performs $m - e - b$ self-transitions into state G and $e - b$ self-transitions into state B.

2. The sequence starts with a burst and ends with a gap, which implies $g = b$. The model performs b transitions from state G to B and $b - 1$ transitions from state G to B. It performs $m - e - g$ self-transitions into state G and $e - g$ self-transitions into state B.

3. The sequence starts with a burst and ends with a burst, which implies $b = g + 1$. The model performs g transitions from state B to G and g transitions from state G to B. It performs $e - g - 1$ self-transitions into state B and $m - e - g$ self-transitions into state G.

4. The sequence starts with a gap and ends with a gap, which implies $g = b + 1$. The model performs b transitions from state B to G and b transitions from state G to

B. It performs $e-b$ self-transitions into state B and $m-e-(b+1)$ self-transitions into state G.

Case 1 is a mirrored representation of case 2 such that both cases occur in equal number. Therefore, the number of sequences of length m that contain e erasures is

$$N(g,b,e,m) = \begin{cases} 2 \cdot \frac{(e-1)!\cdot(m-e-1)!}{(b-1)!\cdot(e-b)!\cdot(b-1)!\cdot(m-e-b)!} & if\ g=b \\ \frac{(e-1)!\cdot(m-e-1)!}{g!\cdot(e-g-1)!\cdot(g-1)!\cdot(m-e-g)!} & if\ b=g+1 \\ \frac{(e-1)!\cdot(m-e-1)!}{(b-1)!\cdot(e-b)!\cdot b!\cdot(m-e-b-1)!} & if\ g=b+1 \end{cases}$$

and their corresponding probability of occurrence is

$$Q(g,b,e,m) = \begin{cases} (1-P_{GG})^b(1-P_{BB})^{(b-1)}P_{GG}^{(m-e-b)}P_{BB}^{(e-b)} & \\ +(1-P_{BB})^g(1-P_{GG})^{(g-1)}P_{GG}^{(m-e-g)}P_{BB}^{(e-g)} & if\ g=b \\ (1-P_{BB})^g(1-P_{GG})^g P_{BB}^{(e-g-1)}P_{GG}^{(m-e-g)} & if\ b=g+1 \\ (1-P_{GG})^b(1-P_{BB})^b P_{GG}^{(m-e-(b+1))}P_{BB}^{(e-b)} & if\ g=b+1. \end{cases}$$

Finally, for $0 \le e \le m$, $P_m(e,m)$ is formulated as

$$P_m(e,m) = \sum_{g=min(1,e)}^{min(e,m-e+1)} \sum_{b=max(g-1,min(1,m-e))}^{min(m-e,g+1)} N(g,b,e,m) \cdot Q(g,b,e,m). \quad (3.47)$$

The complexity of the closed-form analysis can easily be reduced by allocating a limited amount of storage for pre-calculated values. In particular, the factorials appearing in $N(g,b,e,m)$ can be stored in a table of size $O(m)$, where the initial computation requires m multiplications. Further, the powers of the four different transition probabilities in $Q(g,b,e,m)$ are derived via dynamic programming and stored with space complexity $O(4m)$ while requiring a total of $4m$ multiplications for the initial calculation. Given those tables, the closed-form calculation of $P_m(e,m)$ requires up to $4 \cdot \lceil \frac{m}{2} \rceil$ sums and $9 \cdot 4 \cdot \lceil \frac{m}{2} \rceil$ multiplications if the two cases for $g=b$ are calculated separately. Consequently, space and time complexity are linear in the sequence length.

3.3.2. Queueing Analysis

The application of the Gilbert-Elliott model is convenient and has been found to adequately model burst packet loss in wired as well as wireless networks, including erasures due to physical corruption and queue saturation. However, for queueing losses the block error distribution can be derived more accurately in case specific environmental parameters of the network path are known [169]. In particular, such a path is assumed to be limited by a single bottleneck that allocates the minimum throughput to the addressed network flow [20, 54], also known as the network's narrow link. In the case the bottleneck's service rate and the queue length as well as the behavior of the arrivals at the bottleneck are known, a queueing model can be fit in order to simulate the packet loss under the given conditions. This holds especially true under call bandwidth reservation with per-flow network management.

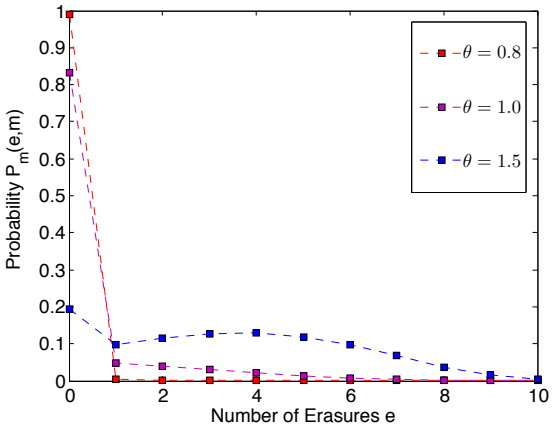


Figure 3.10.: Block-error distribution for a sequence of length $m = 10$ at a single multiplexer with a buffer space of $M = 20$ positions under different saturation. Values from [38].

Queueing models of managed network flows have been extensively studied during the last two decades in order to apply error protection to voice and video flows via adaptive FEC [38, 108, 170]. The following section demonstrates the derivation of the block error distribution for a Markovian queueing model $M/M/1/M^2$ with limited buffer size according to the recursion analysis given by Cidon [38].

Queue Dynamics

The following analysis assumes a network flow that is arriving at a single multiplexer model according to a Poisson process with arrival rate λ. The multiplexer processes arrivals according to a Poisson process with rate μ. The system can store up to M packets in the queue. If the queue has level M at the arrival of a new packet, the packet is discarded. The steady state probability of having $0 \leq j \leq M$ packets in the queue is formulated as P_j in Equation 3.42. The dynamics of packet arrivals and the process of packets being serviced, are described in the following. The probability of a packet arrival to the system before a departure is geometrically distributed [38] with

$$Q_{ad}(\lambda, \mu) = \frac{\lambda}{\lambda + \mu}. \tag{3.48}$$

The probability of a departure before the arrival of a packet is formulated via the complement

$$Q_{da}(\lambda, \mu) = 1 - \frac{\lambda}{\lambda + \mu} = \frac{\mu}{\lambda + \mu} \tag{3.49}$$

Recursion Analysis

In contrast to the Gilbert-Elliott model, the queueing model considers different states of queue level during the arrival of each packet in order to estimate the block error

[2]Kendall's notation: a single queue with exponentially distributed inter-arrival and service times with a limited buffer space for M arrivals.

distribution $P_m(e,m)$. For a queue with a buffer limit M the first packet of a coding block finds the queue at level $0 \leq j \leq M$ with probability P_j (Equation 3.42). Consequently, $P_m(e,m)$ is a superposition of all these cases

$$P_m(e,m) = \sum_{j=0}^{M-1} P_j \cdot P_{m,j+1}(e,m-1) + P_M \cdot P_{m,M}(e-1,m-1), \tag{3.50}$$

where $P_{m,j}(e,m)$ denotes the probability of losing e packets out of a block of length m if the queue level is j during the arrival of the block's first packet. $P_{m,j}(e,m)$ is determined via the following recursion algorithm [38]. The recursion simulates the successive arrival of the m packets of one coding block at the multiplexer's queue. It is initialized with a remaining block length of $m = 1$. The arriving packet is rejected in case the queue is full ($j = M$) and there is no departure before the arrival. Otherwise it is successfully added to the queue:

$$P_{m,M}(e,1) = \begin{cases} \frac{\lambda}{\lambda+\mu}, & e = 1, \\ \frac{\mu}{\lambda+\mu}, & e = 0,\ e \geq 2. \end{cases} \tag{3.51}$$

If at least one buffer position is free, i. e. $0 \leq i \leq M-1$, packet loss cannot occur:

$$P_{m,j}(e,1) = \begin{cases} 1, & e = 0, \\ 0, & e \geq 1. \end{cases} \tag{3.52}$$

For $m \geq 2$ the recursion differentiates between three cases: If the queue is empty upon packet arrival

$$P_{m,0}(e,m) = P_{m,1}(e,m-1) \tag{3.53}$$

or if the queue level is $1 \leq j \leq M-1$

$$P_{m,j}(e,m) = \frac{\lambda}{\lambda+\mu} \cdot P_{m,j+1}(e,m-1) + \frac{\mu}{\lambda+\mu} \cdot P_{m,j-1}(e,m), \tag{3.54}$$

there are no erasures in this recursion step. On the other hand, the packet is lost if the queue is saturated and no departure happens before the arrival of the packet:

$$P_{m,M}(e,m) = \frac{\lambda}{\lambda+\mu} \cdot P_{m,M}(e-1,m-1) + \frac{\mu}{\lambda+\mu} \cdot P_{m,M-1}(e,m) \tag{3.55}$$

The complexity of the recursion is stated as $O\left(Mm^2\right)$, which requires the algorithm to be implemented similarly to the recursion in Section 3.3.1 while storing intermediate results for $P_{m,j}(e,m)$ with space complexity $O\left(Mm^2\right)$.

Figure 3.10 shows the block-error distribution at a single multiplexer under different levels of saturation. The results are obtained during a simulation performed by Cidon et al. [38] that assumes a limited packet queue with drop-tail characteristic in front of a packet multiplexer. The queue has a maximum length of $M = 20$ buffer spaces. A block size of $m = 10$ is simulated. In comparison with Figure 3.9 (left) it becomes obvious that the simplified GE model must assume large correlation coefficients ρ in order to express the high probability of longer erasure sequences e at the multiplexer. This is a result of the GE model's limited number of states in comparison to the queueing model and it holds particularly true in case of a saturated ($\theta = 1$) or over-utilized ($\theta = 1.5$) multiplexer.

4. Protocol Performance Model

"All exact science is dominated by the idea of approximation."
Bertrand Russell

Reliability in modern digital communication goes back to Shannon's fundamental theorem on noisy channel coding. While he formulated a general limit of error-free channel utilization, it has been a major subject in the field of telecommunications research to find codes and protocols that approach the theoretical channel capacity as closely as possible. Interestingly, the applied methodology has been fundamentally different depending on the location of the focussed solution in the overall networking stack. A statement from Chiang et al. formulates it accurately:

"It is well-known that physical layer algorithms try to solve the data transmission problem formulated by Shannon: maximizing data rate subject to the constraint of asymptotically vanishing error probability. Widely used network protocols, such as TCP, BGP[1]*, and IEEE 802.11 DCF*[2]*, were instead designed based primarily on engineering intuitions and ad hoc heuristics."* [31].

A first approach of reliable data transmission over IP infrastructure has been the standardization of the Transmission Control Protocol in 1981. As a result of the intuitive parametrization, the working point of TCP is based on strong assumptions about the network conditions. The protocol performance suffers significantly in presence of large round trip delays and distributed, physical packet losses. A stochastic model of TCP's throughput has been published almost two decades later in form of the TCP response equation by Padhye et al. [116]. The related work has basically been the initiator for equation-based protocol design.

Equation-based design has been recognized as a tool to optimize network congestion control, where the channel utilization is maximized as a function of different network and protocol parameters. Since the predictably reliable protocol combines both proactive and reactive error control under the objective of satisfying application constraints on delay and residual packet loss rate, its channel utilization depends strongly on the interaction of both schemes. Despite being fully configured by the block size k and the repair packet schedule N_P, the protocol spans a large parameter space, which renders an intuitive parameter selection complex.

This chapter develops a stochastic performance model of the predictably reliable protocol, including a reliability model, a feedback model and an efficiency model (Figure 1.1). The analysis composes a complete toolbox for the simulation of the protocol under a variety of application constraints and network states. The reliability model evaluates the protocol's residual packet loss rate. Besides the network path's packet loss rate, the reliability of the feedback delivery has major impact on the protocol performance. In addition, especially in multicast transport the scalability of the receiver feedback is

[1] Border Gateway Protocol
[2] Distributed Coordination Function

crucial. Both effects are simulated within an stochastic feedback model. The efficiency model finally estimates the protocol's overall goodput under a chosen configuration.

4.1. Reliability Model

The fundamental objective of predictable reliability is to explicitly control the residual packet loss rate on transport layer. As the protocol is supposed to operate under strict time constraints, proactive packet repair is required to provide control over the residual loss rate. The adaptive hybrid error coding architecture from Section 2.1.1 relies on algebraic block-erasure codes for the proactive packet repair, which requires careful parametrization prior to the transmission. The correction performance of a protocol parameter set on the current network state must therefore be simulated by the reliability model in order to avoid the transmission of excessive repair packets.

The optimum partitioning of the channel bandwidth between information bits and redundancy bits is a general problem in telecommunications. It has mostly arisen during the design of physical layer coding schemes, where spectral efficiency has been the primary goal. A precondition of suchlike optimization is the availability of a performance model for the applied coding scheme on the corresponding channel model. Especially the performance of algebraic block codes and capacity-approaching randomized codes has been evaluated on Markovian channel models [167, 10]. Previous work covered specifically the performance evaluation of packet-level HEC schemes on packet erasure channels [147, 132].

4.1.1. Residual Erasure Rate

Since the degree of reliability achieved by the predictably reliable protocol on a given network state is a tunable parameter, the analysis needs to decide how many repair packets must be added to the source packet stream. The following block-erasure model determines the residual packet loss rate of the protocol under a certain code rate, i. e. the ratio of source and parity packets in the coding block. The analysis is based on the block error probability as it is obtained from an appropriately chosen block-erasure model (Chapter 3).

Under systematic, packet-level coding, where the coder produces a set of parity packets and appends it to the original set of source packets, it is possible to partially recover the coding block. Those source packets that overcome the network transport are available to the receiving application, even if the block as a whole is not decodable. This effect is considered in the formulation of the residual packet loss probability.

Block Coding Model

The adaptive HEC scheme applies a systematic block-erasure code to k source packets, which produces $n > k$ coded packets, including verbatim copies of the source packets. Initially just the k source packets are transmitted, optionally followed by a small portion $N_p[0]$ of repair packets. The remaining $n - k - N_p[0]$ repair packets are sent reactively upon receiver feedback. In the following it is assumed that the receiver feedback is timely and reliably delivered to the sender. This assumption is later being relaxed by the feedback model in Section 4.2.3, which considers the effect of unreliable loss notifications

on the residual packet loss rate. Under reliable feedback, a residual packet loss implies that the sender has previously sent all n coded packets and less than k of them arrived at the receiver. Since this this is equivalent to the proactive sending of the whole coding block, the residual packet loss rate of the adaptive HEC is modeled as the decoding failure of an (n, k) block-erasure code.

The success of an (n, k) block-erasure code over a network path with packet erasure rate P_e depends on the underlying block-erasure model. Specifically, the block experiences e packet erasures with probability $P_m(e, n)$ according to the block error distribution from Section 3.3. Note that this simplification assumes that packet erasures are independently and identically distributed over all n packets belonging to one coding block. However, due to the fact that the reception of any k packets out of the same block is sufficient to recover from erasures, the location of the erasures within the block is arbitrary for the decoding success. Further, the effect of memory in the erasure channel is also reflected in the block error distribution in that the burstiness increases the probability of a larger number of erasures within the same block.

Partial Decoding

Due to the use of a systematic code, the receiver benefits from all successfully received source packets. Therefore, the residual packet loss rate considers all cases of decoding failure and counts the number of unrecovered source packets within those cases. A decoding failure refers to the cases where less than k arbitrary, coded packets of the same block are available at the receiver.

Let a block of n coded packets, including k source packets, experience j erasures, which occurs with probability $P_m(j, n)$. Within the block, up to k source packets might be lost as well as up to $n-k$ parity packets. Under the assumption that the j erasures are independently and identically distributed within the coding block, source packets and parity packets share the erasures according to the hypergeometric distribution $P_d(n, k, i, j)$:

$$P_d(n, k, i, j) \quad = \quad \frac{\binom{k}{i}\binom{n-k}{j-i}}{\binom{n}{j}}. \tag{4.1}$$

Specifically, the distribution denotes the probability that j erasures in a block of length n affect i among the k source packets.

Whether the i source packets are finally lost, depends on the overall decoding success of the block. If not more than $i \leq n-k$ source packets are erased, a decoding failure happens only if at the same time at least $j \geq n-k+1$ packets are lost overall. On the other hand, if the number of erasures among the source packets exceeds already the number of parity packets sent, i. e. $i > n-k$, the decoding is not possible, independently from the number of parity packets received. Cases in which all erasures concentrate on the parity packets do not contribute to the residual packet loss rate. Therefore, a residual loss occurs if at least one source packet is erased ($i \geq 1$) and overall at least $j = max(n-k+1, i)$ packets are erased within the coding block. The residual packet erasure rate is the average number of unrecovered source packets per block under all cases of decoding failure, which is formulated as

$$P_r(k, n) \quad = \quad \frac{1}{k} \sum_{i=1}^{k} \sum_{j=max(n-k+1,i)}^{n-k+i} i \cdot P_d(n, k, i, j) \cdot P_m(j, n). \tag{4.2}$$

Given a constraint P_T for the residual erasure rate of a specific application, Equation 4.2 is used to tune n for a given k and a given block error distribution $P_m(e, n)$ such that $P_r(k, n)$ meets the constraint.

4.1.2. Incremental Redundancy

Under the cooperation of block-erasure coding and packet repair requests the parity packets are sent in periodic repair cycles, whereas each cycle is initiated by the corresponding receiver feedback. The repair process terminates if either no more repair cycles are allowed (e. g. due to a delay constraint) or if the previous repair cycle is successful. This coding scheme delivers redundancy incrementally as far as it is required under the measured network state (Section 2.1.1), which automatically optimizes the code rate. Therefore, the number of coded packets that actually cross the network is quantized by the repair packet schedules of each repair cycle and results in a specific aggregate block length after each cycle (Figure 2.1). The residual packet loss probability after each repair cycle is a crucial parameter for the protocol efficiency since it determines the initiation of the subsequent repair packet schedule. Therefore, the probability of decoding failure is determined after each cycle.

Aggregate Block Length

Upon receiving a loss notification from the receiver, the sender sends $N_P[c]$ parity packets in repair cycle $0 \leq c \leq N_C$. Therefore, the incremental block length for a specific protocol configuration is denoted by a vector with $N_C + 1$ positions. The *aggregate block length* $n[c]$ models the overall number of coded packets that are sent after successful completion of repair cycle c including the k source packets. As $n[c]$ is formulated from the sender's point of view, successful completion of the cycle relies only on the assumption that the loss notification of the receiver has reached the sender:

$$n[c] = k + \sum_{b=0}^{c} N_P[b] \tag{4.3}$$

with $0 \leq c \leq N_C$. Given the aggregate block length $n[c]$, the residual erasure probability is obtained after each repair cycle c via Equation 4.2 as $P_r(k, n[c])$.

Probability of Decoding Failure

The aggregate block length models the incremental redundancy scheme from the sender's point of view. At the receiver the number of packets received after each repair cycle is, however, subject to the network's packet erasure probability. Let A_c be a random variable for the number of packets aggregated at the receiver after repair cycle c. Let a_c be an instance of A_c indicating the number of packets received out of the aggregate block length $n[c]$ that has been sent. Then $n[c] - a_c$ packets are lost during the transmissions up to cycle c and the probability mass function of A_c is formulated based on the block error distribution $P_m(e, m)$ as follows:

$$Pr(A_c = a_c) = P_m(n[c] - a_c, n[c]), \tag{4.4}$$

where $0 \leq a_c \leq n[c]$.

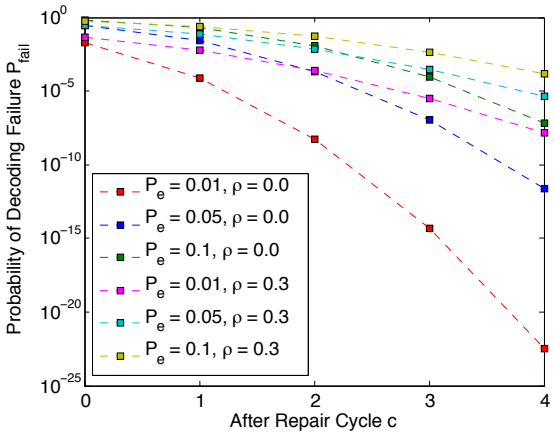


Figure 4.1.: Probability of decoding failure $P_{fail}(c)$ after a number of repair cycles c under the protocol configuration $k = 20$, $N_P = [1, 2, 3, 4, 5]$.

The decoding success of a repair cycle c is determined by the event of aggregating at least k coded packets at the receiver. Thus, the receiver must not lose more than $j = n[c] - k + 1$ packets out of $n[c]$. The probability of decoding failure $P_{fail}(c)$ after repair cycle c is formulated via the cumulative distribution of A_c being less than k. As a failure of the previous cycle $c - 1$ is a precondition for the activation of the repair cycle c, $P_{fail}(c)$ is formulated as a conditional probability:

$$\begin{aligned} P_{fail}(c) &= Pr(A_c < k) \cdot P_{fail}(c-1) \\ &= \left(\sum_{j=n[c]-k+1}^{n[c]} P_m(j, n[c]) \right) \cdot P_{fail}(c-1). \end{aligned} \tag{4.5}$$

for $0 < c \leq N_C$, whereas

$$P_{fail}(0) = Pr(A_0 < k) = \sum_{j=n[0]-k+1}^{n[0]} P_m(j, n[0]). \tag{4.6}$$

Figure 4.1 shows the probability of decoding failure after all repair cycles of the protocol configuration $k = 20$, $N_P = [1, 2, 3, 4, 5]$. P_{fail} decays roughly exponentially with the number of cycles. The correlation coefficient as well as the protocol configuration (e. g. the relationship between the block length k and the chosen repair packet schedule N_P) determine the exponent of the decay.

4.1.3. Model Accuracy

The accuracy of the reliability model is of particular interest since a predictable residual packet loss rate is the fundamental objective of the developed protocol. Further, the reliability model has influence on both the feedback model and the efficiency model since loss notifications and repair packets are issued upon a residual packet loss in a previous repair cycle. As the computational complexity is an issue in adaptive systems, the reliability model introduces simplifying assumptions. The statistical simplifications

as well as their impact on the model's accuracy are discussed in the following section. Simulations show that the assumptions generally lead to a conservative estimation of the residual packet loss rate.

Impact of the Simplifications

Block-erasure performance models from earlier work [10, 147, 167] have been developed for simulation purposes. However, their complexity prevents their application in real, adaptive systems. Therefore, the available reliability model builds upon two major simplifications. First, the sequence analysis that models the distribution of packet erasures within the coding block, is relaxed so as to neglect the actual position of the erasures within the block. This simplification allows the interface between the block-erasure model and the protocol performance model to be reduced to the block error distribution, which enables the performance model to cooperate with various block-erasure models. As a result, the reliability model performs a purely quantitative analysis considering the number of erasures per coding block but not their actual distribution within the block. In fact, it is difficult to model the exact network state during the transmission of a single packet under a high degree of multiplexing in the network. Therefore, the block error distribution is applied to the block as a whole, where the erasures are assumed to be distributed over source and parity packets according to a hypergeometric distribution (Equation 4.1).

As a second simplification, the temporal distance of subsequent packets of the same block is not considered during the statistical analysis of the residual packet loss rate, which is a result of the block coding model (Section 4.1.1). Hence, both proactive and reactive transmission of repair packets are modeled similarly and the influence of temporal correlation in the packet loss of subsequent packets is neglected. Nevertheless, the distribution of the repair packets over multiple transmission cycles that are reasonably longer than one packet interval leads to interleaving of source and parity packets of different coding blocks, which results in significant de-correlation of the packet loss process for the respective coding block. Therefore, under high temporal correlation the correction performance of short block codes is considerably higher under reactive repair compared to proactive repair. In simulation models, the distance between two subsequent repair cycles can be interpolated by a number of transitions of the Markovian block-erasure model in order to characterize the interleaving effect of reactive packet repair at the price of significantly increasing computational complexity [147]. The neglected interleaving effect leads to a conservative estimation of the residual packet loss rate. Temporal correlation in the packet loss process translates into a larger accumulation of erasures within the same block, which is reflected by the block error distribution.

Simulation Results

The accuracy of the reliability model is evaluated against a simulation of the adaptive HEC coding architecture. In each simulation experiment 10^8 source packets are sent with a specific protocol configuration (block length k and repair packet schedule N_P). During the following experiments, the duration of one repair cycle is assumed to correspond to 10 source packet intervals. The simulation runs on top of a simplified GE model, configured with packet loss probability $P_e = 0.05$ and correlation coefficient ρ. The reliability model as well as the simulations express the protocol's residual packet loss rate without respect

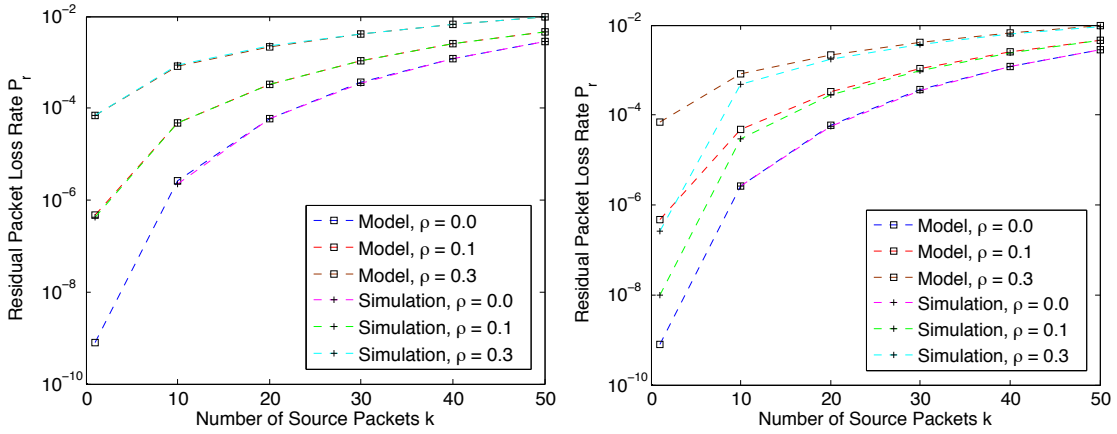


Figure 4.2.: Accuracy of the reliability model under purely proactive repair ($N_P = [6]$, left) and reactive repair ($N_P = [0, 1, 2, 3]$, right).

to timing errors. Timing errors contribute to the residual loss rate if protocol deadlines are missed by any packet due to transmission delays or scheduling latencies at the end hosts. The protocol performance model, however, purely models the statistical success of the error control, based on the assumption that the timing model cares for the estimation of sufficient delay margins.

Figure 4.2 compares the residual packet loss rate predicted by the reliability model with the simulation of proactive and reactive protocol configurations. During the first experiment, the repair packet schedule specifies the sending of six repair packets proactively (left graph: $N_P = [6]$). Afterwards it is spread over three reactive repair cycles (right graph: $N_P = [0, 1, 2, 3]$). The number of repair packets is kept constant while the number of source packets per block increases from $k = 1$ to $k = 50$, which results in increasing code rate. Therefore, the experiment produces residual packet loss rates within different orders of magnitude.

The first experiment shows that the relaxation of the sequence analysis does not affect the model's accuracy. Model and simulation match perfectly for the purely proactive protocol configuration, even under a large correlation coefficient. The second experiment evaluates the model under temporal interleaving of the repair packets due to reactive repair. In contrast to the simplified block coding model (Section 4.1.1), the repair packets of different cycles are spaced away by 10 source packet intervals during the simulation. Whereas the model provides still perfect accuracy under uncorrelated packet loss, it deviates from the simulation for short block lengths ($k < 10$) in presence of large correlation coefficients. For a block length of $k = 1$, it overestimates the residual packet loss rate by up to two orders of magnitude under a correlation coefficient of $\rho = 0.3$. This effect results from the neglect of the temporal distance between adjacent packets of the same block. However, due to the underestimated distance the model overestimates the impact of the temporal loss correlation such that it delivers a conservative prediction of the residual packet loss rate. Model and simulation converge quickly towards each other for protocol configurations with more than 10 source packets per block. The effect is explained as follows. The block error distribution models the probability of an erasure length less or equal than the overall coding block length ($k + \sum_{c=0}^{N_C} N_P$), regardless of their spacing. Under large correlation coefficients the distribution expects longer sequences of erasures

with higher probability. If the number of source packets per block is larger than the number of parity packets, the probability of an initial loss of a longer sequence among the continuously transmitted source packets dominates the residual packet loss rate, such that the model improves in accuracy for larger k.

4.2. Feedback Model

Adaptive error control is not possible without frequent updates from the receiver about the perceived reception quality. However, the receiver feedback of a large multicast group aggregates to a considerable amount of bandwidth and hampers the scalability. In fact, frequent receiver feedback is likely to cause *feedback implosions* at the sender [110]. This effect results in a breakdown of the service since either the sender or the network infrastructure is overwhelmed by the flood of receiver messages. Finding the optimal trade-off between adaptivity and feedback bandwidth is therefore the primary issue of reliable multicast schemes, which has been reflected in the definition of appropriate *feedback suppression* schemes.

On the other hand, the loss of feedback messages affects the performance of reactive error control schemes. Under the usage of negative acknowledgments, it results in a failed packet repair cycle because the sending of repair information is not triggered. Therefore, the effect of *unreliable feedback* has major impact on the residual packet loss rate of NACK-based protocols. However, the return path is usually assumed to be error-free in the development of reactive error control [132, 147]. The following feedback model introduces a comprehensive analysis including the effects of both feedback suppression and unreliable feedback on the reliability model of the predictably reliable protocol.

4.2.1. Feedback Bandwidth

Standards in the IP-based media multicast either limit the bandwidth of the receiver feedback by statistical means [114] or even deny receiver feedback at all [53]. As an example, RTP/AVPF allows sender and receivers to occupy not more than 5% of the session bandwidth for their protocol-specific reports, while sender and receiver reports should share the feedback bandwidth in a ratio of 1:3. The predictably reliable protocol specifies two types of feedback: periodic updates of the network state and loss notifications. Whereas the bandwidth of the network state feedback is conveniently controlled by a specific interval, the bandwidth of the loss notifications depends on code parameters as well as the network's packet loss rate.

Network State Feedback

Network state feedback contributes periodic updates on essential network parameters in order to feed the network monitoring at the sender. The sending interval is independent from coding and erasure events on the network. In order to formulate the bandwidth share of the network state feedback, a fixed feedback interval of T_{NFB} is assumed. The network state feedback scales linearly with the number of receivers N_R in a receiver group. Given a specific feedback packet size L_{NFB}, the network state feedback causes an average rate of

$$R_{NFB} = \frac{N_R \cdot L_{NFB}}{T_{NFB}}. \tag{4.7}$$

In general, scalability in multicast sessions is obtained by limiting the bandwidth occupied by the receiver feedback [26, 114]. Under a given receiver group size the feedback interval T_{NFB} is tuned in order to meet a specific bandwidth constraint. However, a larger T_{NFB} reduces the sampling frequency of the network monitoring in larger multicast groups.

Loss Notifications

Loss notifications indicate an insufficient number of repair packets at the beginning of a repair cycle. For each block, loss notifications are sent at a particular interval D_{REQ} that is specified by the sender in order to compensate for the communication delay in each repair cycle. The maximum number of loss notifications for one coding block is limited by the number of repair cycles N_C.

The following feedback model is based on the assumption that packet erasures occur with equal probability and are not spatially correlated among different receivers. Therefore, the probability of decoding failure $P_{fail}(c)$ after repair cycle c is assumed to be i. i. d. over the N_R receivers. Let F_c be a random variable representing the number of receivers that need to send a loss notification in cycle c. Under the above assumptions, F_c is binomially distributed with mean

$$E[F_c] = N_R \cdot P_{fail}(c-1). \tag{4.8}$$

A loss notification has packet size L_{LFB}. An expected number of $E[F_c]$ notifications is sent from the whole receiver group in repair cycle c. For each coding block, i. e. the time interval of k source packets appearing with interval T_S, the protocol specifies N_C repair cycles. Therefore, the average feedback bandwidth generated by the receiver group's loss notifications yields

$$R_{LFB} = \frac{\sum_{c=1}^{N_C} E[F_c] \cdot L_{LFB}}{k \cdot T_S}. \tag{4.9}$$

4.2.2. Feedback Suppression

With scaling receiver groups the feedback traffic consumes a reasonable part of the bandwidth. However, especially in case of the loss notifications, a significant number of the feedback messages contributes overlapping information. This is for instance an effect of spatially correlated packet loss, which causes the feedback to arrive at the multicast source in synchronized bursts. In order to avoid redundant feedback by exhausting overlapping information in feedback messages, various feedback suppression methods that preserve the scalability of error control in large multicast groups have been developed [164, 114, 132, 110, 13, 22].

The related work in the field of reliable multicast has developed mainly two types of feedback suppression schemes [164]: timer-based suppression and the suppression based on representatives. Timer-based feedback suppression works probabilistically by introducing a randomly chosen suppression timer. While waiting for an individual timeout, each receiver delays the own feedback and listens to the other receiver's feedback in order to determine redundant information. In the other approach, representatives are determined based on their observed network state. Ideally, the representative is the receiver that experiences the worst-case network state.

Timer-based feedback suppression fits particularly well into the timing model of the predictably reliable protocol. As feedback messages are already scheduled via absolute timers, they can be delayed by a random timer while the protocol's timing model compensates for the additional delay. In the following the resulting feedback quantity is obtained under timer-based suppression, based on the analysis in [110].

Timer-based Feedback Suppression

Timer-based feedback suppression relies on the assumption that all receivers of the multicast group listen to each other's feedback. If a receiver observes a loss notification from another receiver of the group, it checks whether the contained information is redundant with respect to its own feedback. Loss notifications are redundant as soon as they both address the same coding block as well as the same repair cycle. Under this condition the receiver suppresses its own feedback upon reception of the foreign loss notification. In order to increase the probability for receivers to receive foreign feedback before the sending of the own feedback, timer-based feedback suppression specifies random timers. The local feedback is delayed by the chosen timer while the receiver is listening for redundant feedback.

In the following, a model is constructed for the effect of timer-based feedback suppression on the predictably reliable error control scheme (Figure 4.3). Let $D_{SUP}[c]$ be a delay margin for the feedback suppression in repair cycle c. Without loss of generality, consider two receivers r and s that schedule a loss notification for the same block in a multicast group. In repair cycle c both receivers randomly choose a timer from a timer distribution with density $f_T(T)$ over the interval $[0, D_{SUP}[c]]$. r's feedback is being suppressed if s chooses a timer T_s that is smaller than the timer T_r of r by at least the forward trip time $FTT_{r,s}$ between both: $0 \leq T_s + FTT_{r,s} < T_r \leq D_{SUP}$.

$D_{SUP}[c]$ is added to the c^{th} repair request timer in order to compensate for the case that the maximum timer is being chosen. Therefore, Equation 2.11 is reformulated under timer-based feedback suppression to

$$D_{REQ}[c] = D_{SUP}[c] + RTT + N_p[c] \cdot D_{PTx} + D_{RS}. \tag{4.10}$$

As the residual loss rate decreases exponentially with an increasing number of repair cycles (Section 4.1.2), the number of loss notifications emitted by the group reduces significantly. Therefore, it might be reasonable to shorten the timer interval for later repair cycles.

Feedback Quantity

In order to evaluate the efficiency of the timer-based feedback suppression, the feedback quantity after the suppression must be formulated. Let the random variable F_r denote the success of the feedback suppression at receiver r. $F_r = 0$ corresponds to the cases in which the receiver suppresses the own feedback because of the reception of redundant feedback. $F_r = 1$ indicates that the feedback is sent.

Let overall N_{FB} instances of redundant feedback be generated among the group. The own feedback is suppressed at receiver r if at least one receiver s chooses a timer such that its feedback arrives at receiver r before T_r expires. Therefore, the feedback is sent with the joint probability of the following events: Receiver r chooses timer T_r and any

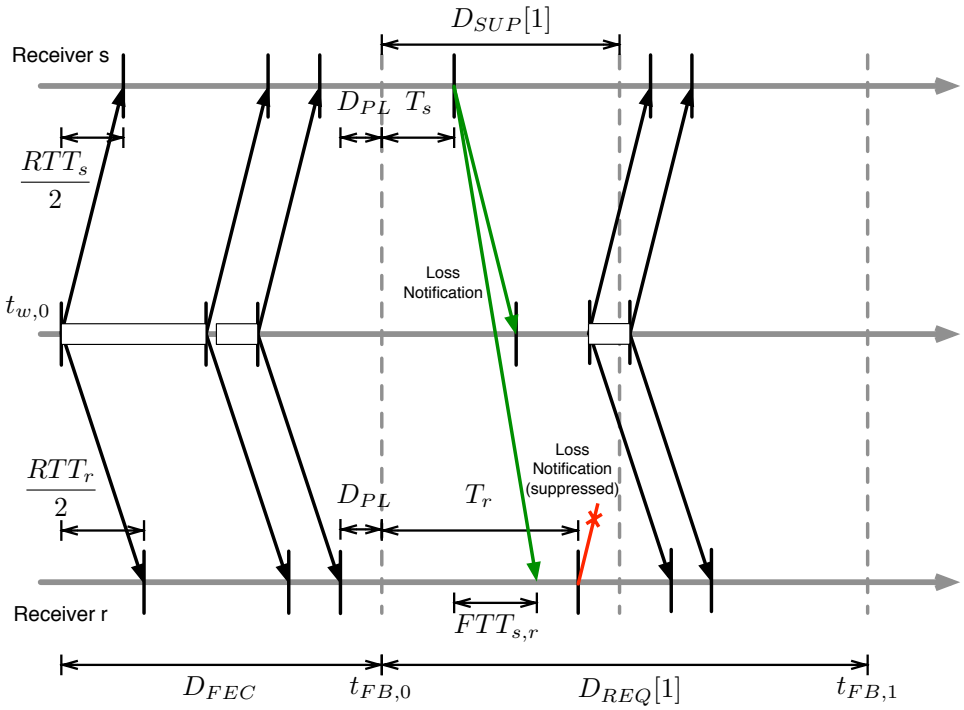


Figure 4.3.: Timer-based feedback suppression within the predictably reliable protocol.

other receiver $s \neq r$ chooses a timer that is greater or equal than $T_r - FTT_{r,s}$. F_r has the following probability density function [110]:

$$Pr(F_r = 1) = \int_0^{D_{SUP}} f_T(T_r) \prod_{s=1, s\neq r}^{N_{FB}} \left(\int_{T_r - FTT_{r,s}}^{D_{SUP}} f_T(T_s)\, dT_s \right) dT_r. \tag{4.11}$$

Timer Distribution

The distribution of the suppression timer determines the efficiency of the feedback suppression. In the following the residual feedback quantity is derived at the example of a uniformly distributed timer. Nonnenmacher calculates the expected number of feedbacks sent based on different distributions of $f_T(T)$ [110]. The residual number of feedbacks can be formulated based on any of these distribution functions in a similar fashion. Let $f_T(T)$ be defined in the domain $[0, D_{SUP}]$ as follows:

$$f_T(T) = \begin{cases} \frac{1}{D_{SUP}}, & T \in [0, D_{SUP}] \\ 0, & otherwise \end{cases}. \tag{4.12}$$

Plugging Equation 4.12 into Equation 4.11 yields the following probability for receiver r to send its feedback:

$$Pr(F_r = 1) = \int_0^{D_{SUP}} \frac{1}{D_{SUP}} \prod_{s=1, s\neq r}^{N_{FB}} \left(\int_{T_r - FTT_{r,s}}^{D_{SUP}} \frac{1}{D_{SUP}} dT_s \right) dT_r. \tag{4.13}$$

Intuitively, if the FTT between both receivers exceeds the maximum suppression timer, the feedback is sent deterministically, i. e. $FTT_{r,s} \geq D_{SUP} > 0$

$$Pr(F_r = 1) = \int_0^{D_{SUP}} \frac{1}{D_{SUP}} \prod_{s=1, s \neq r}^{N_{FB}} 1 dT_r = 1. \tag{4.14}$$

If the suppression timer can compensate for the FTT, i. e. $0 < FTT_{r,s} < D_{SUP}$, receiver r's feedback is sent in two cases: Firstly, the suppression fails if the timer of r is already less than the FTT to receiver $s \neq r$, i. e. $T_r < FTT_{r,s}$, which happens with probability

$$\begin{aligned} Pr(F_r = 1) &= \frac{1}{D_{SUP}} \int_0^{FTT_{r,s}} \prod_{s=1, s \neq r}^{N_{FB}} 1 dT_r \\ &= \frac{FTT_{r,s}}{D_{SUP}}. \end{aligned} \tag{4.15}$$

Secondly, if the timer of r is not greater than the timer of $s \neq r$ by at least the FTT between both i. e. $T_r - FTT_{r,s} < T_s \leq D_{SUP}$. This happens with the following probability:

$$\begin{aligned} Pr(F_r = 1) &= \frac{1}{D_{SUP}} \int_{FTT_{r,s}}^{D_{SUP}} \prod_{s=1, s \neq r}^{N_{FB}} \left(\frac{1}{D_{SUP}} \left(D_{SUP} + FTT_{r,s} - T_r \right) \right) dT_r \qquad (4.16) \\ &= \frac{1}{D_{SUP}^{N_{FB}}} \left[\left(-\frac{FTT_{r,s}^{N_{FB}}}{N_{FB}} \right) - \left(-\frac{D_{SUP}^{N_{FB}}}{N_{FB}} \right) \right] \\ &= \frac{1}{N_{FB}} \left[1 - \left(\frac{FTT_{r,s}}{D_{SUP}} \right)^{N_{FB}} \right]. \end{aligned}$$

Approximation of the Feedback Quantity

Let the group round trip time (GRTT) be known as the maximum RTT between the sender and any receiver of the multicast group. It is assumed that any two receivers have a maximum one-way propagation delay of no more than one GRTT, i. e. $GRTT \geq max_{r,s} FTT_{r,s}$. This refers to the situation that both receivers are centered around the sender with maximum distance. Let $Pr(F_r = 1)$ be calculated under the assumption that the one-way communication between any two receivers is equally delayed by one GRTT, i. e. $FTT_{r,s} = GRTT$ for $1 \leq r, s \leq N_{FB}$. This approximation results in an upper bound for $Pr(F_r = 1)$ as the success of the feedback suppression increases with smaller $FTT_{r,s}$. The expected number of feedbacks sent from the whole group after feedback suppression is therefore approximated by

$$E[F_{sup}] = \sum_{r=1}^{N_{FB}} E[F_r] = N_{FB} \cdot Pr(F_r = 1). \tag{4.17}$$

Replacing $Pr(F_r = 1)$ as obtained from Equation 4.13 into Equation 4.17 leads to the expected quantity of feedbacks under the uniform timer distribution:

$$E[F_{sup}] = \begin{cases} N_{FB}, & GRTT \geq D_{SUP} > 0 \\ 1 + \frac{GRTT \cdot N_{FB}}{D_{SUP}} - \left(\frac{GRTT}{D_{SUP}} \right)^{N_{FB}}, & 0 < GRTT < D_{SUP} \end{cases}. \tag{4.18}$$

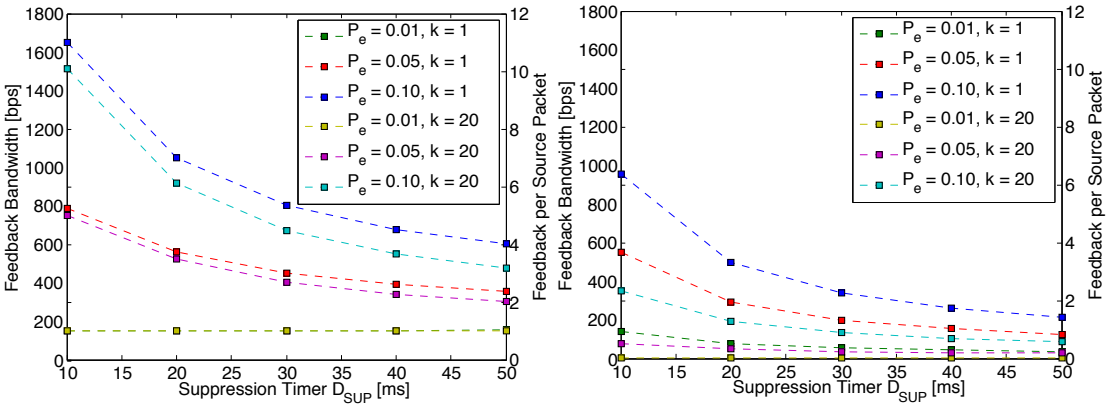


Figure 4.4.: Feedback Bandwidth for different suppression timers D_{SUP} for 100 receivers and $RTT = 10\,ms$. Protocol parameters: $k = 1$, $N_p = [0, 1, 2, 3]$ (left); $k = 20$, $N_p = [0, 2, 4, 8]$ (left); $k = 1$, $N_p = [0, 2, 4]$ (right); $k = 20$, $N_p = [2, 2, 4, 6]$ (right).

For large group sizes the expression is further approximated by

$$E[F_{sup}] \approx \frac{GRTT}{D_{SUP}} \cdot N_{FB}. \tag{4.19}$$

If an individual suppression interval $D_{SUP}[c]$ is assigned to each repair cycle c, the expected number of loss notifications in cycle c is

$$E[F_c] = \frac{GRTT}{D_{SUP}[c]} \cdot N_R \cdot P_{fail}(c-1) \tag{4.20}$$

after feedback suppression. The residual feedback bandwidth is obtained by replacing $E[F_c]$ in Equation 4.9.

The residual feedback bandwidth and the number of feedbacks sent after feedback suppression are depicted in Figure 4.4 for a multicast group with 100 receivers. Besides longer suppression timers, mainly two effects reduce the resulting feedback bandwidth: longer coding blocks as well as an increased proactivity in sending repair packets.

4.2.3. Unreliable Feedback

The reliability of the feedback delivery depends strongly on the underlying network infrastructure. Feedback can be assumed to be totally reliable in wired local and wide area networks as long as the return path is not congested. In wireless network environments, however, it is lost with high probability as a result of collisions in the shared medium access. The problem of packet loss in the feedback channel has been addressed by Rubenstein for hybrid erasure coding [132]. However, an analytical modeling of the feedback path is missing in his work. In the following such a feedback loss model is derived for the adaptive HEC scheme that is integrated into the predictably reliable protocol. The model leads to a refinement of the residual erasure probability P_r (Section 4.1.1) and considers the effect of repeated loss notifications as suggested by Rubenstein [132] as well as the impact of the multicast group size on the feedback reliability.

Feedback Erasure Model

Unreliable feedback turns reactive error correction into a two-stage stochastic experiment, where both the sending of incremental parity and the sending of feedback are statistically dependent. Intuitively, the loss of feedback increases the probability of residual packet loss since the sender omits the transmission of the corresponding repair packet schedules. In a naively implemented reactive error control scheme, a lost feedback message in repair cycle c corresponds to leaving out the portion of repair packets $N_P[c]$. However, under the assumption that the sender can recognize lost feedback for any cycle $b < c$ in cycle c with $1 < c \leq N_C$, it can compensate for the lost feedback by sending the repair packets for all previous cycles at once, i. e. sending $\sum_{b=0}^{c} N_P[b]$ packets in cycle c. Therefore, the sender is able to recover from the loss of feedback in earlier cycles as soon as a later feedback comes in. This is achieved by signaling c or a feedback sequence number in the feedback packet.

In case the sender compensates for lost feedback in the above fashion, no repair cycle is omitted up to the last feedback received. Intuitively, the incremental redundancy is truncated after the repair cycle corresponding to the last successful feedback reception. If the last feedback reaches the sender in cycle $c < N_C$, this results in a truncated aggregate block length $n[c] < n[N_C]$ composed of k source packets and $\sum_{b=0}^{c} N_P[b]$ repair packets. Obviously, there are $N_C + 1$ alternatives for the overall code word length sent, depending on the success of the feedback transmission.

Residual Erasure Rate

The block coding model of Section 4.1.1 is extended in order to formulate the residual packet erasure probability under unreliable feedback. In particular, the model's assumption that residual packet loss is preceded by the sending of the entire amount of available repair packets is adopted, i. e. all potential repair cycles are used. This assumption removes the statistical dependence of the feedback erasure probability from the probability of decoding failure after each repair cycle. In contrast to the original block coding model, the repair cycle requested within the last successfully delivered feedback determines the overall number of repair packets sent. This value is therefore being assumed for the overall code word length n in the refined block coding model.

Let A_{ufb} be a random variable denoting the aggregate block length sent after repair cycle c under unreliable feedback, i. e. $A_{ufb} \in \{n[c] \mid 0 \leq c \leq N_C\}$ (compare Section 4.1.2). Further, let $P_{mfb}(A_{ufb} = n[c], P_f, N_C)$ be the probability mass function of A_{ufb} under i. i. d. feedback erasure probability P_f and N_C repair cycles. According to the above feedback loss model, the aggregate block length $A_{ufb} = n[c]$ is being sent in case at least cycle c's feedback has been received and all later loss notifications are erased.

Two special cases need to be considered: $n[0]$ packets are sent for $c = 0$ if the feedback of all N_C reactive cycles is lost, which happens with probability $Pr\left(A_{ufb} = n[0]\right) = P_f^{N_C}$. On the other hand, $n[N_C]$ packets are sent if the last feedback is received, i. e. $Pr\left(A_{ufb} = n[N_C]\right) = 1 - P_f$. The general case $1 \leq c < N_C$ is geometrically distributed with the number of truncated repair cycles. If the cycle index of the final feedback received is c, there is a tail of $N_C - c$ unused repair cycles denoted by $P_f^{(N_C - c)}$. At the same time exactly the c^{th} feedback is successful with probability $1 - P_f$. Due to the recovery from lost feedback at the sender, the success of the feedback before the c^{th} cycle

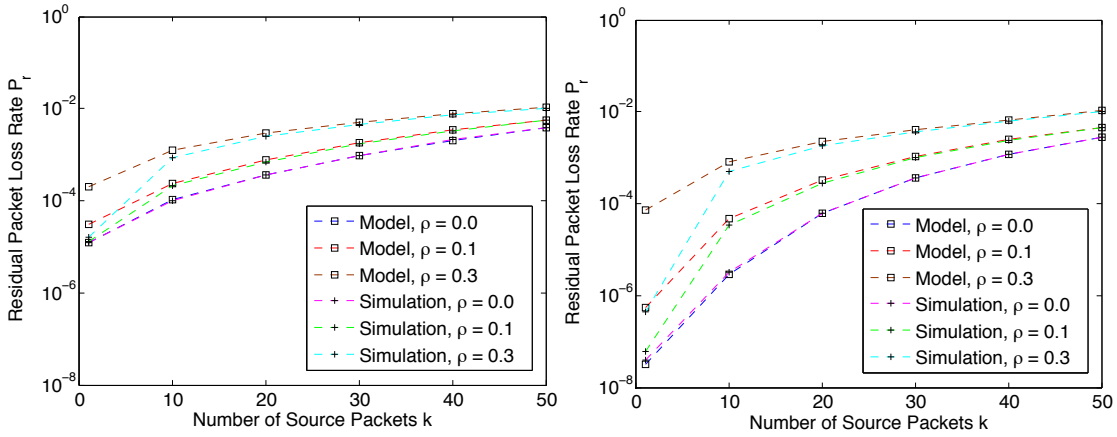


Figure 4.5.: The impact of unreliable feedback on the residual packet loss rate and the accuracy of the reliability model under unreliable feedback without (left) and with (right) repetition of the last cycle's loss notification ($d = 3, P_f = P_e$).

is arbitrary. Finally, $P_{mfb}(n[c], P_f, N_C)$ is obtained as

$$\begin{aligned} P_{mfb}(n[c], P_f, N_C) &= Pr\left(A_{ufb} = n[c]\right) \\ &= \begin{cases} P_f^{N_C}, & c = 0 \\ P_f^{(N_C - c)}\left(1 - P_f\right), & 1 \leq c < N_C \\ 1 - P_f, & c = N_C \end{cases}. \end{aligned} \tag{4.21}$$

Based on the assumed statistical independence of the feedback erasure probability P_f, the overall residual packet loss rate under packet erasure probability P_e is formulated using the block coding model from Section 4.1.1:

$$P_{r,ufb}(k, n, P_f) = \sum_{c=0}^{N_C} P_r\left(k, n[c]\right) \cdot P_{mfb}\left(n[c], P_f, N_C\right). \tag{4.22}$$

The left graph of Figure 4.5 shows the effect of unreliable feedback on the residual packet loss rate. The experiment repeats the accuracy evaluation from Section 4.1.3 with the reactive protocol configuration while the feedback loss rate P_f is set as high as the network path's original packet loss rate P_e. In comparison with Figure 4.2 (right) it becomes obvious that the impact of lost feedback is particularly pronounced for low packet loss rates. However, during the experiment it vanishes with larger P_e and larger k, which are situations in which the packet loss rate in the forward path dominates the result.

Feedback Repetition

In the above model the reception of the last cycles's feedback dominates the decision whether feedback loss has an impact on the residual erasure rate, i. e. the full repair packet schedule is sent with probability $Pr\left(A_{ufb} = n[N_C]\right) = 1 - P_f$. Therefore, the reliability of the reactive erasure coding scheme increases significantly if the last cycle's

feedback is repeated d times. Under repetition of the last feedback, the maximum aggregate block length $n[N_C]$ is sent with probability $Pr\left(A_{ufb} = n[N_C]\right) = 1 - P_f^d$ if the erasure probability of the repeated feedback is assumed to be i. i. d. Consequently, the probability mass function $P_{mfb}(n[c], P_f, N_C, d)$ of the aggregate block length A_{ufb} sent under d repetitions of the last cycles's feedback is refined to

$$\begin{aligned} P_{mfb}(n[c], P_f, N_C, d) &= Pr\left(A_{ufb} = n[c]\right) \\ &= \begin{cases} P_f^{(N_C-1+d)}, & c = 0 \\ P_f^{(N_C-c-1+d)}\left(1 - P_f\right), & 1 \leq c < N_C. \\ 1 - P_f^d, & c = N_C \end{cases} \end{aligned} \tag{4.23}$$

Intuitively, the repetition factor d decreases the probability of the geometrically distributed length of the tail of unsuccessful repair cycles exponentially. It can be tuned in order to achieve the successful delivery of the last cycle's feedback with a certain probability. With this probability the maximum aggregate block length $A_{ufb} = n[N_C]$ is sent. Let P_{rf} be a chosen reliability level for the last cycle, then the repetition factor is determined as follows:

$$\begin{aligned} & P_{rf} &&= P_{mfb}\left(n[N_C], P_f, N_C, d\right) \\ \Leftrightarrow \quad & P_{rf} &&= 1 - P_f^d \\ \Leftrightarrow \quad & d &&= \left\lceil \frac{log\left(1 - P_{rf}\right)}{log\left(P_f\right)} \right\rceil . \end{aligned} \tag{4.24}$$

As depicted in the right graph of Figure 4.5, a small repetition factor of $d = 3$ for the last cycle's loss notification can significantly compensate for the unreliable feedback such that the residual packet loss rate is close to that under reliable feedback (compare Figure 4.2, right).

Influence of the Group Size

A larger receiver group size causes natural repetition of feedback due to two reasons. Under high spatial correlation in the packet loss process, several receivers experience the erasure of the same packets with high probability. Further, with longer block size, the probability of multiple receivers requiring repair packets for the same coding block increases even under an independent packet erasure process. Consequently, an increasing group size causes redundant loss notifications and increases the reliability of feedback. Section 4.2.1 defines F_c as a random variable for the number of loss notifications sent in repair cycle c. According to Equation 4.8, a group of N_R receivers produces in average $E[F_c] = N_R \cdot P_{fail}(c)$ loss notifications in cycle c. In case feedback suppression is applied, $E[F_c]$ is obtained from Equation 4.17.

Under the assumption that the feedback erasure probability P_f is equal for each receiver in the multicast group, the probability mass function $P_{mfb}(n[c], P_f, N_C, d, N_R)$ of the aggregate block length A_{ufb} under feedback repetition factor d in a multicast group of

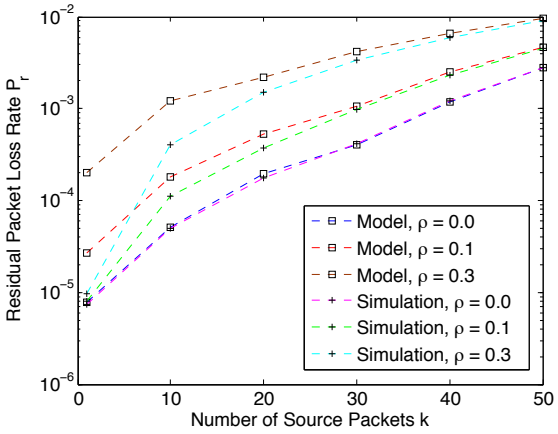


Figure 4.6.: The residual packet loss rate under unreliable feedback and a group size of $N_R = 30$ ($d = 1$).

size N_R is refined to

$$P_{mfb}(n[c], P_f, N_C, d, N_R) = Pr\left(A_{ufb} = n[c]\right) \tag{4.25}$$
$$= \begin{cases} \prod_{b=1}^{N_C-1} P_f^{E[F_b]} \cdot P_f^{(E[F_{N_C}]\cdot d)}, & c = 0 \\ \prod_{b=c+1}^{N_C-1} P_f^{E[F_b]} P_f^{(E[F_{N_C}]\cdot d)} \left(1 - P_f^{E[F_c]}\right), & 1 \le c < N_C \\ 1 - P_f^{(E[F_{N_C}]\cdot d)}, & c = N_C \end{cases}.$$

In General it is beneficial to reduce or disable the feedback suppression for later repair cycles in order to increase the overall feedback reliability. As shown in Figure 4.6, a group size of $N_R = 30$ receivers already increases the overall reliability of the protocol similarly to the feedback repetition factor (compare Figure 4.5) due to the presence of redundant feedback if feedback suppression is switched off.

4.3. Efficiency Model

Whereas the overall number of repair packets scheduled for transmission determines the residual packet loss rate of the adaptive HEC (Section 4.1.1), their distribution among the available repair cycles has a strong effect on the efficiency of the predictably reliable protocol. Other than an FEC scheme, which deterministically adds a fixed number of parity packets in order to achieve the desired code rate, the HEC shares the available bandwidth between source and repair data stochastically depending on the network state. The efficiency model calculates the expected amount of repair data – in the following referred to as *redundancy information* – that is added to the source stream under a certain protocol configuration. In particular, the protocol's goodput under a given bandwidth limit is predicted under consideration of the expected redundancy information and the overall messaging overhead.

4.3.1. Packet-level Redundancy

The packet-level redundancy determines the per-packet coding overhead of a protocol parameter set. The redundancy information RI is expressed as a fraction of the source

data rate. The stream rate after the error control is $R_S \cdot (1 + RI)$ without consideration of packet headers.

Expected Redundancy Information

The redundancy information on the network path depends on the distribution of the repair packets among the N_C repair cycles as well as the amount of proactive repair packets $N_P[0]$. If receiver feedback is assumed to be reliable, the sender transmits $n[c] = k + \sum_{b=0}^{c} N_P[b]$ packets up to and including cycle $1 \leq c \leq N_C$. The repair cycle c is triggered by a negative acknowledgment from the receiver if less than k packets are received after cycle $c-1$, which occurs with probability $P_{cycle}(c) = P_{fail}(c-1)$ (Equation 4.5). The expected amount of redundancy required by one receiver is obtained as

$$RI(k, N_C, N_P) = \frac{1}{k} \cdot N_P[0] + \frac{1}{k} \cdot \sum_{c=1}^{N_C} P_{cycle}\,(c) \cdot N_P[c]. \tag{4.26}$$

In case of multicast transport, $RI(k, N_C, N_P)$ depends on the packet loss behavior of each individual receiver. Let packet erasures occur statistically independent at N_R receivers in a multicast group. Let $P_{m,r}(e, m)$ be the block error distribution determined on receiver j's network state observations, where $1 \leq r \leq N_R$. Similarly to Equation 4.5, let $P_{fail,r}(c)$ be the probability that cycle c fails at receiver r. The repair packets for cycle c are sent as soon as at least one receiver sends a loss notification. Therefore, $P_{cycle}(c)$ is refined as follows for the case of multicast transmission:

$$P_{cycle}\,(c) = 1 - \prod_{r=1}^{N_R} \left(1 - P_{fail,r}(c-1)\right). \tag{4.27}$$

However, this equation would require to maintain a separate block-erasure model for each receiver at the multicast sender in order to calculate $P_{fail,r}(c)$ based on the block error distribution $P_{m,r}(e, m)$ of the r^{th} receiver. As the related bookkeeping overhead at the sender severely affects the scalability of the model, a conservative approximation of $P_{cycle}(c)$ is obtained as follows. Let $P_{fail,max}(c)$ represent the maximum probability of decoding failure on cycle c within the receiver group. As no receiver observes a higher failure rate than $P_{fail,max}(c)$, the probability $P_{cycle}\,(c)$ of applying cycle c in a multicast group with N_R receivers is at most

$$P_{cycle}\,(c) \leq 1 - \left(1 - P_{fail,max}(c-1)\right)^{N_R}. \tag{4.28}$$

$P_{fail,max}$ is obtained by applying the reliability model on the block-erasure model derived from the packet loss observations of a group representative, which usually represents the worst-case conditions among the group.

Efficiency of reactive Erasure Coding

The reactive repair of packet erasures has significant effect on the efficiency of the adaptive HEC. Intuitively, the coding scheme increases in efficiency as the number of repair cycles increases and repair packets are shifted to later cycles. This is a result of the exponentially decreasing failure probability of the block-erasure coding (Figure 4.1, Equation

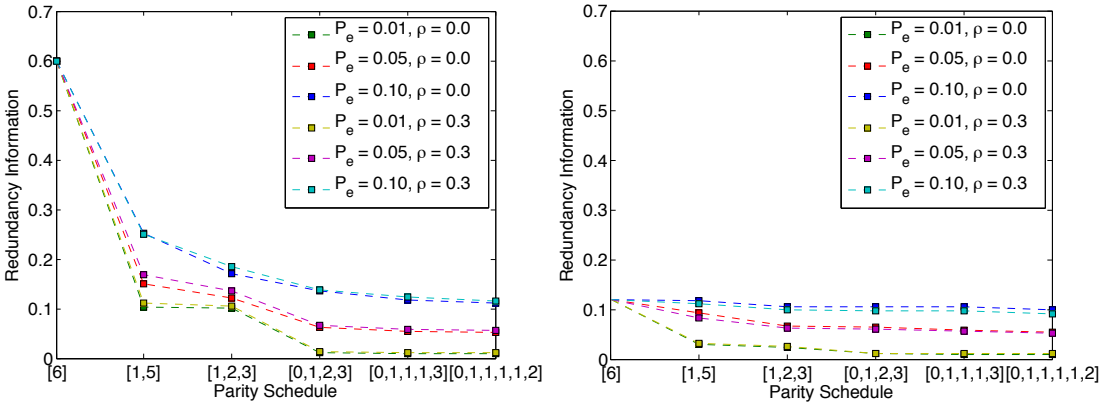


Figure 4.7.: Impact of reactive packet repair. Block length $k = 10$ (left) and $k = 50$ (right).

4.5) in the expression of the expected redundancy information (Equation 4.26). However, the number of reactive repair cycles is limited under tight delay constraints or large communication delays.

Figure 4.7 compares the required redundancy information for different configurations, where six repair packets are either sent proactively or they are distributed over one to five reactive repair cycles. The coding block length is set to $k = 10$ (left) and $k = 50$ (right), respectively. Throughout all experiments the adding of the first reactive repair cycle ($N_P = [6]$ to $N_P = [1, 5]$) leads to the most significant increase in coding efficiency since starting from this point, the majority of the repair packets is just added in case actual erasures occur on the network path. Especially for short block lengths, entirely switching off the proactive repair ($N_P = [1, 2, 3]$ to $N_P = [0, 1, 2, 3]$) leads a second step of decreasing redundancy (Figure 4.7, left). For large block sizes under high erasure rates the distribution of the repair packets is widely arbitrary (Figure 4.7, left). In the considered experiment the temporal correlation of the packet loss process has minor impact on the coding overhead of the HEC scheme. This effect results from the fixed parameterization of the protocol during the experiment, which leads to a higher residual packet loss rate under larger correlation coefficients.

As the number of repair cycles increases, the required redundancy information tends towards the theoretical optimum as defined in (Section 3.1.2, Equation 3.12). This effect is clearly visible for all graphs in Figure 4.7. For instance, the configuration $k = 10$ with 6 repair packets has a residual packet loss rate of $P_r = 2.2497 \cdot 10^{-4}$ on a network path with erasure rate $P_e = 0.1$. In this case, the optimal redundancy information is stated by Equation 3.12 as $RI_{opt} = \frac{0.1-P_r}{1-0.1} = 0.1109$, which is the asymptote of the respective graphs in the left figure.

Impact of the Receiver Group Size

Packet erasures tend to be spatially independent in large receiver groups unless the path to several receivers is affected by a common error source. Protocol configurations with short coding block lengths, especially repetition codes with $k = 1$, lose significantly efficiency with an increasing group size as they must handle the independent packet losses individually. In the worst case the effort for the multicast source to answer loss

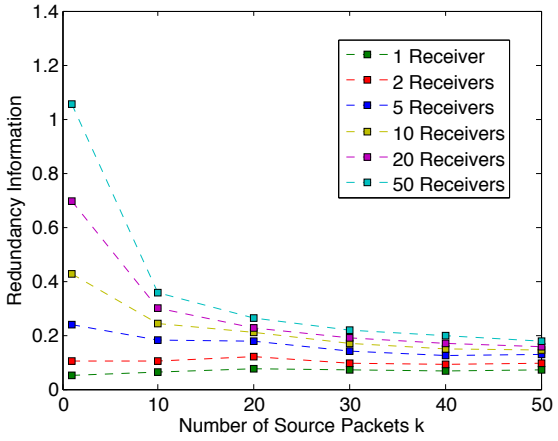


Figure 4.8.: Impact of the receiver group size ($P_e = 0.05$, $\rho = 0$).

notifications from the receiver group scales linearly with the group size. The forward error coding component within the adaptive HEC generates universal repair information for the spatially uncorrelated packet erasures since within the same coding block a parity packet can replace any source packet during the decoding process. The longer the coding blocks of the HEC, the more loss notifications can potentially be answered with the same portion of repair packets.

The redundancy information that is required under different coding block sizes k is evaluated for different receiver group sizes in Figure 4.8. Each receiver is assumed to independently lose packets with erasure probability $P_e = 0.05$. The number of repair packets is adapted for each chosen k in order to achieve an equal residual packet loss rate. Further, the repair schedule is being optimized under a fixed number of repair cycles $N_C = 3$. Therefore, the graphs express the pure effect of the block size and the receiver group size on the redundancy information under a constant reliability requirement. For 50 receivers the coding overhead exceeds the source rate ($RI > 1$) under a block length of $k = 1$. Already a block length of $k = 10$ requires less than half of the redundancy information for the same group size. However, longer coding blocks decrease in efficiency on small multicast groups. This is a result of the fact that configurations with longer blocks require a certain degree of proactivity in the repair packet schedule, which amortizes for larger receiver groups. In the given scenario, block lengths beyond $k = 1$ become feasible for group sizes of more than three receivers. Yet for small receiver groups the redundancy decreases asymptotically as the coding blocks become significantly longer (e. g. for $k > 20$ in the given example).

Model Accuracy

It becomes evident in Section 4.1.3 that the simplifications in the reliability model lead to an overestimation of the residual packet loss rate for short block lengths ($k < 10$) under correlated packet loss. Correspondingly, the probability of decoding failure (Equation 4.5) is overestimated. Figure 4.9 evaluates the accuracy of the efficiency model under the same conditions as in Figure 4.2, i. e. a repair packet schedule of $N_P = [0, 1, 2, 3]$ is defined for an increasing block length k while reactive transmissions are assumed to have a temporal distance of 10 source packet intervals.

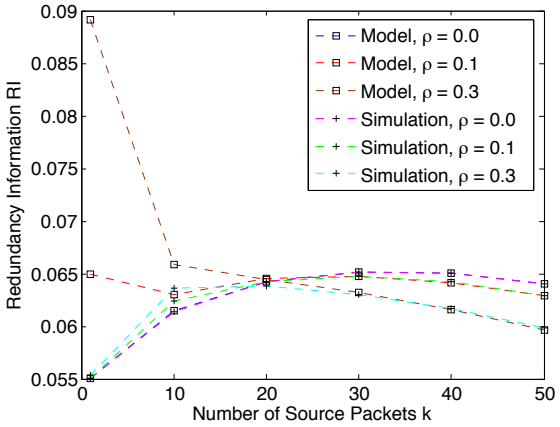


Figure 4.9.: Accuracy of the efficiency model. Protocol configuration: $N_P = [0, 1, 2, 3]$

The conservative estimation of the residual packet loss rate for short coding blocks under temporal correlation causes an overestimation of the required redundancy information for those configurations. Similarly to the reliability model, the efficiency model reproduces the simulation perfectly for longer coding blocks. The estimation of the redundancy information is a crucial step in the optimization of protocol parameters under the objective to maximize the protocol efficiency (Section 5.1.1). The overestimation of the redundancy information penalizes configurations with short coding blocks during the protocol optimization on network paths with highly correlated packet loss. This turns out to be a beneficial feature as longer coding blocks have de-correlating effect on the bursty packet erasures.

4.3.2. Bandwidth Utilization

Block-erasure coding requires the position of the erasure to be known at the receiver. Error detection and erasure localization are therefore tasks of the transport protocol. In addition, time stamps as well as coding parameters must be included into source and repair packets, respectively, in order to support the adaptively implemented block-erasure coding in the predictably reliable protocol. Loss notifications and network state feedback must be transmitted on the return path from the receiver to the sender at the same time. Altogether those protocol features require additional header fields to be added to the protocol datagrams.

Header Overhead

The protocol's header overhead is not included in the estimation of the redundancy information (Equation 4.26) as it depends on the actual protocol design. In the following the protocol goodput is calculated under consideration of the header and feedback overhead as well as the redundancy information. This analysis is crucial in case the protocol operates under a rate limit such as imposed by a congestion control algorithm. An accurate estimation of the protocol goodput in practical transmission scenarios requires a model of the protocol's header overhead. PRRT's header format is specified in Section 2.1.3.

Let source packet headers be of length L_{DHdr} and repair packet headers of length

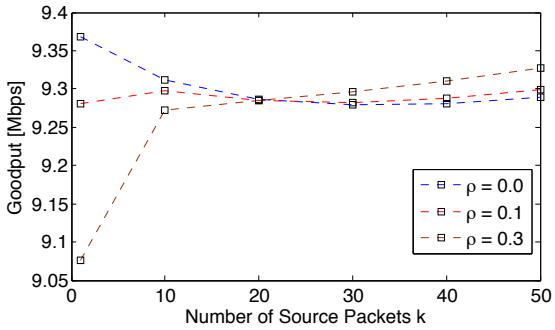


Figure 4.10.: Protocol goodput under a bandwidth limit of 10 *Mbps* and a packet loss rate of $P_e = 0.05$. Protocol configuration: $N_P = [0, 1, 2, 3]$ ($L_{DHdr} = 24\,byte$, $L_{PHdr} = 16\,byte$, $L_D = 1316\,byte$, $L_P = 1368\,byte$).

L_{PHdr}. Furthermore, let L_D be the payload length of a source packet and L_P the payload length of a repair packet, respectively. Equation 4.26 formulates the per-packet redundancy information, i. e. a fraction of RI repair packets transmitted per source packet. Consequently, at a source rate R_S, the header overhead is

$$R_{PO} = R_S \cdot \left(\frac{L_{DHdr}}{L_D} + \frac{RI \cdot L_{PHdr}}{L_P} \right). \tag{4.29}$$

Goodput Calculation

The protocol's effective throughput R_{eff} is composed of the redundancy information RI of the applied protocol configuration (k, N_C, N_P), the rates of loss notifications R_{LFB} and network state feedback R_{NFB} as well as the header overhead R_{PO}. At a source rate R_S, it yields

$$R_{eff}\,(RI, R_S) \quad = \quad (1 + RI) \cdot R_S + R_{PO} + R_{LFB} + R_{NFB}. \tag{4.30}$$

The protocol goodput is defined as the source rate carried by the protocol after deduction of coding, feedback and header overhead. Given a rate limit $R_C (= R_{eff})$, the equation is solved for R_S in order to determine the source rate that is transmitted with a residual packet loss rate of $P_r(k, k + \sum_{c=0}^{N_C} N_P)$:

$$R_S\,(RI, R_C) = \frac{R_C - R_{LFB} - R_{NFB}}{\left(1 + RI + \frac{L_{DHdr}}{L_D} + \frac{RI \cdot L_{PHdr}}{L_P}\right)}. \tag{4.31}$$

If the protocol feedback is sent over a separate path, R_{LFB} and R_{CFB} should be set to zero in the goodput calculation.

Figure 4.10 shows the protocol goodput achieved during the experiment in Section 4.1.3 under a bandwidth limit of 10 *Mbps*. The following values are chosen for the protocol's message format: length of the source frame header $L_{DHdr} = 24\,byte$, length of the repair packet header $L_{PHdr} = 16\,byte$, source packet payload size $L_D = 1316\,byte$ and repair packet payload size $L_P = 1328\,byte$. The feedback path is assumed to be separate. As reflected in Equation 4.31, the goodput behaves basically reciprocally to the redundancy

information. Therefore, it also shares the efficiency model's discrepancy for short block lengths under bursty packet loss. However, this leads to reservation of more bandwidth headroom for configurations with short coding blocks and purely reactive repair schedules under a large correlation coefficient. This result is beneficial in that such configurations have a much higher peak-to-average bandwidth requirement since several short blocks in sequence might be hit by an error burst such that all of them activate the reactive error control at the same time. Longer coding blocks reduce the peak-to-average bandwidth as their repair packets amortize for a larger number of source packets and the expected number of blocks affected by the error burst is considerably lower. This effect is also visible in Figure 4.10. For longer source blocks the goodput increases under higher correlation because packet erasures tend to be rather concentrated within a low number of coding blocks instead of being equally distributed over the whole packet stream.

5. Predictably Reliable Error Control

"We cannot direct the wind, but we can adjust the sails"
Bertha Calloway

In general, the design of transport layer protocols follows a modular architecture that separates essential functional blocks such as error control and congestion control. In modern network calculus, these modules are described via mathematical control equations with proven stability such that the protocol's throughput can be stated as a function of the observed network state. Those concepts are concluded under the term network decomposition, which understands each layer of the network stack as a decomposed solution to the optimization problem of digital channel utilization [31].

Reliable network protocols require closed-loop error control, which relies on receiver notifications about the success of the repair process. Besides the fact that total reliability cannot be achieved in open-loop error control, this design principle is also widely chosen because of the dynamic channel capacity in packet-switching networks. Reactive and hybrid repair schemes determine the optimum code rate under such conditions due to their inherent adaptivity. Transport layer error control schemes strive for zero residual error rate as they are commonly designed for total reliability such that a specific, non-zero set point does not exist for the residual error rate.

Throughout this chapter, a control system for predictably reliable error control is being developed in order to efficiently approach a residual target error rate for loss-tolerant network content. The controller predicts protocol parameters by applying the stochastic protocol performance model from Chapter 4 on the current network state. Thereby, it optimizes the parametrization so as to achieve the desired reliability under the objective of maximum channel utilization. The optimization problem as well as a fast search algorithm are formulated in the following. In addition, a potential method for the selection of a representative network state estimate in a heterogeneous multicast group is proposed.

Since error control increases the bandwidth requirement of continuous real-time network traffic, the scheme must consider the existence of an upper bandwidth limit. The bandwidth limitation may be obtained by an explicit setting or via equation-based congestion control. In particular, the interaction of equation-based congestion control and error-controlled media transport is being discussed in this chapter.

5.1. Protocol Parameter

Following the goal of digital channel coding – the maximization of spectral efficiency – it is desirable to find the protocol configuration that fulfills the application constraints while minimizing the redundancy transmitted over the network path. The corresponding optimization problem is defined in the following section. Whereas application constraints can be assumed to be static over time, characteristics of the network path, such as available bandwidth, packet erasure rate and round trip time, are subject to transient changes. Since those are essential input parameters of the optimization problem, protocol parameters require frequent updates under consideration of the channel's coherence time.

The complexity of the optimization algorithm is a significant issue in real implementations. The large number of input parameters as well as the variability of the output parameters renders off-line policies ineffective due to large storage requirements. Therefore, the following section formulates some heuristics into a greedy search in order to enable the on-line calculation of coding parameters. A fast search algorithm is presented that finds the global optimum of the protocol parameter space under the protocol's timing model (Section 2.2) and the modeled packet loss characteristics (Section 3.2).

5.1.1. Optimization Problem

The predictably reliable protocol is explicitly configured by the number of source packets per coding block as well as the corresponding schedule of repair packets. Both parameters are mutually dependent. Each parameter set operates under a specific combination of coding delay, residual packet loss probability and coding efficiency as determined by the protocol's timing and performance models. A feasible parameter set is determined by the comparison of the predicted protocol performance with the application constraints. This corresponds to the solution of a combinatorial optimization problem in which the optimum solution is chosen among a feasible set of discrete parameter settings.

Input Variables

The input variables are subject to transient changes of application settings or network conditions. The input parameter space is updated by the protocol's network state monitoring and the related modeling. It includes the following variables:

- **Block Error Distribution $P_m(e, m)$:** Depending on the applied block-erasure model (Section 3.2), the packet loss probability and a correlation coefficient describing the burstiness of the packet erasure process are being estimated. The block-erasure model is represented via its specific block error distribution (Section 3.3).
- **Round Trip Delay RTT:** The protocol samples the RTT from each bidirectional communication event between sender and receiver. The RTT includes the network's propagation and queueing delays.
- **Available Bandwidth R_C:** For the parameter optimization it is assumed that the available bandwidth of the network is known. The protocol throughput is strictly limited by the available bandwidth such that it has significant impact on the feasibility of a protocol parameterization. The available bandwidth might be estimated via a congestion control equation (Section 5.3).

- **Average Source Rate R_S:** The average source rate of the content carried by the protocol is measured at any time. The source rate is essential for the calculation of the protocol throughput, i. e. the effective rate R_{eff} of the coded packet stream. Besides, it determines the average packet interval T_S, which is an important parameter of the protocol's timing model.

- **Receiver Group Size N_R:** The number of receivers significantly influences the efficiency of the protocol. It is estimated via a bookkeeping mechanism in the protocol by counting the active sources of receiver feedback.

Application Constraints

The optimization problem is formulated under the following application-specific constraints:

- **Delay Constraint D_T:** The application delay constraint defines the time budget available for the end-to-end delivery of each single transmission unit. A data unit must not be delayed by more than D_T between entering the network stack at the sender and delivery to the receiver application.

- **Erasure Probability Constraint P_T:** The protocol must adjust the number of repair packets dynamically such that it achieves the target reliability that is required by the application. Constraints for the residual erasure probability are usually defined as a long-term average in the scale of one or several hours.

- **Rate Constraint R_T:** The estimate of the available bandwidth R_C is interpreted as an additional constraint R_T. In case R_T puts a limitation on the amount of redundancy that is added by the protocol to a given source rate R_S, the residual erasure probability is not necessarily limited by P_T.

Output Variables

The protocol is explicitly parameterized by the following variables:

- **Coding Block Length k:** The coding block length determines the number of source packets in the block-erasure code. It is variable due to short-term variations in the source rate but limited by either k or the number of source packets collected within the maximum coding delay D_C (Equation 2.7).

- **Number of Repair Cycles N_C:** As a result of the measured communication delay, a certain number of reactive repair cycles is possible under the delay constraint D_T.

- **Repair Packet Schedule N_P:** The repair packet schedule determines the distribution of the repair packets resulting from the block coding over the $N_C + 1$ repair cycles, including the proactive repair packets. N_C is implicitly determined by the length of N_P minus one.

Objective Function

Each triple (k, N_C, N_P) representing a combination of the output variables under consideration of their individual ranges defines a valid protocol parameter set. In order to determine the optimum protocol configuration under the application constraints, the protocol parameter set is evaluated via the objective function. The optimum configuration refers to the parameter triple (k, N_C, N_P) that fulfills the time constraint D_T, the rate constraint R_T as well as the reliability constraint P_T at the same time and minimizes the objective function.

The objective of the optimization is to minimize the coding overhead under the current network state and the application constraints. This corresponds to finding the parameter combination (k, N_C, N_P) that requires the globally minimum redundancy information RI_{min} as follows (Equation 4.26):

$$RI_{min} = \min_{(k,N_C,N_P)} RI\,(k, N_C, N_P) \tag{5.1}$$

subject to (Equation 2.9 and Equation 2.11)

$$D_{FEC} + N_C \cdot D_{REQ} \quad \leq \quad D_T \tag{5.2}$$

and (Equation 4.2)

$$P_r\,(k, n) \leq P_T,\ with\, n = k + \sum_{c=0}^{N_C} N_P[c] \tag{5.3}$$

and (Equation 4.30)

$$R_{eff}\,(RI(k, N_C, N_P), R_S) \leq R_T. \tag{5.4}$$

These three constraints, however, interfere mutually. For instance, a particular number of repair packets might be required under a chosen coding block length k in order to achieve the desired level of reliability as denoted by P_T. A larger number of repair packets, on the other hand, increases the effective rate R_{eff} of the coded packet stream. In addition, the redundancy information increases with a stricter delay constraint D_T since the granularity of feasible protocol parameter sets increases and the protocol must assign more repair packets to earlier repair cycles (Figure 4.7) such that R_{eff} might exceed the bandwidth constraint R_T. Since R_T is a physical constraint, the reliability constraint P_T is relaxed in the following, which virtually corresponds to the effect of an over-utilized communication channel under the channel coding theorem. If the bandwidth constraint is too tight so as to reach the desired residual erasure probability, the optimization problem translates into minimizing the residual erasure rate under D_T and R_T:

$$P_{r,min} = \min_{(k,N_C,N_P)} P_r\,(k, n)\,,\ with\, n = k + \sum_{c=0}^{N_C} N_P[c] \tag{5.5}$$

subject to

$$D_{FEC} + N_C \cdot D_{REQ} \quad \leq \quad D_T \tag{5.6}$$

and

$$R_{eff}\,(RI(k, N_C, N_P), R_S) \leq R_T. \tag{5.7}$$

The configuration that minimizes the residual packet loss rate under the bandwidth constraint corresponds still to the configuration that requires the minimum amount of redundancy. Therefore, the objective of optimum coding efficiency remains valid under the relaxed reliability constraint.

Distribution of the Time Budget

The optimization problem understands the application's delay constraint D_T as a limited time budget that is shared by proactive and reactive packet repair operations. As a result of the parameter optimization, the optimum distribution of D_T among both mechanisms is determined. Unfortunately, the overall optimization problem formulates several recursive dependencies, e. g. between the block length k and the number of repair cycles N_C as well as the repair request delay D_{REQ} and the repair packet schedule N_P (Equation 5.1 to Equation 5.7 using Equation 2.9 and Equation 2.11). Therefore, those parameters must either be adjusted iteratively in suitable quantization steps according to their dependencies or they are left as manual parameters to be defined beforehand.

In order to resolve the recursive dependency with respect to the repair packet schedule N_P, a maximum number $n_{P,max}$ of repair packets per cycle is defined as a manual parameter. Further, in case feedback suppression is activated, the maximum suppression timer D_{SUP} must be added to D_{REQ} in each cycle. Otherwise set $D_{SUP} = 0$. Afterwards, D_{REQ} (Equation 2.11) depends only on input variables as follows:

$$D_{REQ} = RTT + n_{P,max} \cdot D_{PTx} + D_{RS} + D_{SUP}. \tag{5.8}$$

Each block must include at least one source packet. Therefore, a minimum coding delay $D_{FEC,min}$ (compare Equation 2.9) is deducted from the delay budget D_T:

$$D_{FEC,min} = T_S + n_{P,max} \cdot D_{PTx} + \frac{RTT + D_{RS}}{2} + D_{PL}. \tag{5.9}$$

According to Section 2.3, D_{PL} is set to $4.5 \cdot T_S$ in order to differentiate between erasures and out-of-order packets. Given $D_{FEC,min}$ and D_{REQ}, $N_{C,max}$ is obtained by quantizing the remaining delay budget in steps of size D_{REQ}. $N_{C,max}$ is the maximum number of reactive repair cycles:

$$N_{C,max} = \left\lfloor \frac{D_T - D_{FEC,min}}{D_{REQ}} \right\rfloor. \tag{5.10}$$

For any N_C, $0 \leq N_C \leq N_{C,max}$ the specific maximum block length $k(N_C, D_T)$ is determined after deducting the delay of N_C repair cycles from the time budget D_T:

$$\begin{aligned} D_T &\geq D_{FEC} + N_C \cdot D_{REQ} \\ &\geq k \cdot T_S + n_{P,max} \cdot D_{PTx} + \frac{RTT + D_{RS}}{2} + D_{PL} + N_C \cdot D_{REQ} \\ k\left(N_C, D_T\right) &= \left\lfloor \frac{D_T - n_{P,max} \cdot D_{PTx} - \frac{RTT + D_{RS}}{2} - D_{PL} - N_C \cdot D_{REQ}}{T_s} \right\rfloor. \end{aligned} \tag{5.11}$$

The maximum code word length n_{max} of the block-erasure code additionally limits the coding block length $k(N_C, D_T)$. The value of n_{max} depends on the type and the parameterization of the erasure code. For instance, under a symbol size of 8 bit[1] the Vandermonde erasure code (Section 2.3) supports a maximum code word length of $n_{max} = 255$.

[1] In network erasure coding, broadcast and consumer electronics, block-erasure codes usually operate on bytes, i. e. the elements of the $GF(2^8)$, which is convenient for hardware processing.

Larger code symbols and equivalently larger n_{max} increase the data volume of the coding block in order to support media sources of very high data rate. These configurations are obtained if the code is defined on a Galois field of higher order. However, $n_{max} = 255$ allows for a sufficiently large k under typical application delay constraints and media source rates (see further: measurement scenarios in Chapter 6). In addition, the volume of a coding block can conveniently be adjusted via the protocol's payload length while keeping the symbol size constant.

Given a specific n_{max}, $k(N_C, D_T)$ is upper-bounded by the minimum code rate required to satisfy the reliability constraint. Therefore, let k_{lim} be the maximum coding block length k that satisfies $P_r(k, n_{max}) \leq P_T$. Consequently, the block length is limited by $k_{max}(N_C, D_T) = min(k(N_C, D_T), k_{lim})$ in order to allow a sufficient number of parity packets to be produced during the encoding process.

5.1.2. Search Algorithm

Due to the large number of input variables, the search space of the parameter optimization has significant size. In the following the size of the full search space is determined and a fast search algorithm is being developed that approaches the global optimum of the result space by early rejection of infeasible and suboptimal parameter sets. A greedy algorithm is formulated that generates the feasible space while following the gradient of the objective function. In addition, an interpolation method is proposed that approximates a good initial solution for Equation 5.3, i. e. the number of repair packets n required under a certain coding block length k.

Search Space

The general search algorithm comprises three steps. First, all feasible distributions of the time constraint are generated by quantizing D_T with integer multiples of the repair request delay D_{REQ} (Equation 2.11 and Equation 5.10). Therefore, all combinations of the number of repair cycles N_C and the corresponding coding block length $k(N_C, D_T)$ are possible with $0 \leq N_C \leq N_{C,max}$ and $1 \leq k \leq k_{max}(N_C, D_T)$ (Equation 5.11). Second, for each individual k the N_C+1 repair cycles must incrementally be filled with repair packets, either until the reliability constraint is met or as long as the bandwidth constraint is not exceeded. Hence, each generated repair packet schedule is jointly being evaluated via the reliability model (Equation 4.2) and the efficiency model (Equation 4.26). Finally, if the feasible space with respect to D_T, R_T and P_T is non-empty, the solution with the minimum redundancy information is determined. Otherwise, the parameter set with the minimum residual packet loss rate is chosen from the feasible space subject to D_T and R_T.

Consider a feasible distribution of the time budget with N_C repair cycles and a block length k. Let $n_{k,opt}$ be the maximum code word length of the block-erasure code that satisfies the bandwidth constraint with at least one potential repair packet schedule or alternatively, the minimum code word length that fulfills $P_r(k, n_{k,opt}) \leq P_T$ for k. This parameter is limited by the erasure code to $n_{k,opt} \leq n_{max}$. The algorithm must test all pairs (k, n) with $k < n \leq n_{k,opt}$ in order to find $n_{k,opt}$. For each pair (k, n) the $n - k$ repair packets are delivered over the proactive and the N_C reactive repair cycles. Each repair cycle can be filled with up to $n_{P,max}$ repair packets.

$C(8,3,1,5) = 18$					$P(8,3,1,5) = 4$
1, 2, 5	2, 1, 5	2, 5, 1	3, 4, 1	5, 1, 2	1, 2, 5
1, 3, 4	2, 2, 4	3, 1, 4	4, 1, 3	5, 2, 1	1, 3, 4
1, 4, 3	2, 3, 3	3, 2, 3	4, 2, 2		2, 2, 4
1, 5, 2	2, 4, 2	3, 3, 2	4, 3, 1		2, 3, 3

Table 5.1.: Integer compositions and integer partitions of 8 with 3 parts with minimum size 1 and maximum size 5.

Recall that $N_P[0]$ repair packets are assigned to the proactive repair cycle. The problem of distributing the remaining $r = n - k - N_P[0]$ repair packets among the N_C reactive cycles corresponds to the generation of all *integer compositions* [73] of r. The integer compositions of r define the set of all $C(r) = 2^{r-1}$ possibilities of writing r as a sum of positive integers. In particular, *restricted integer compositions* are generated under the specification of additional constraints such as the number of parts in the sum as well as the minimum and maximum size of all parts. Let $C(r, s, a, b)$ denote the number of restricted integer compositions of r with s parts of minimum size a and maximum size b. Restricted integer compositions can be obtained via generating functions or recursion formulas [113], which also determine their number. For illustration, Table 5.1 enumerates the $C(8, 3, 1, 5) = 18$ integer compositions of 8 with 3 parts, where $a = 1$ and $b = 5$.

Given an assignment for n, k and $N_P[0]$, the parameter search space contains $C(r, N_C, 1, n_{P,max})$ options to deliver the $r = n - k - N_P[0]$ reactive repair packets among N_C cycles, where each cycle must schedule at least one packet and not more than $n_{P,max}$. The proactive repair cycle is either empty or it sends up to $n_{P,max}$ repair packets such that $0 \leq N_P[0] \leq n_{P,max}$. Consequently, the entire search space has a size of

$$\sum_{N_C=0}^{N_{C,max}} \sum_{k=1}^{k_{max}(N_C)} \sum_{n=k+1}^{n_{k,opt}} \sum_{N_P[0]=0}^{min(n_{P,max},\, n-k-N_C)} C\left(r,\, N_C,\, 1,\, n_{P,max}\right), \tag{5.12}$$

with $r = n - k - N_P[0]$.

Reduced Search Space

A simple heuristic proposed by Tan [147] leads to a reduced size of the search space due to coarser sampling of the coding block length k. The heuristic is based on the assumption that the objective function with respect to the redundancy information (Equation 5.1) is a concave function of k. Hence, the most efficient configuration of the hybrid coding architecture is either already obtained by a pure repetition code ($k = 1$) or, in case of longer block-erasure coding, with the maximum possible code word length, which optimally amortizes the parity symbols of the block-erasure code. This assumption is indeed supported by the concave shape of the redundancy graph in Figure 4.9 except for the efficiency model's deviation under large correlation coefficients. Under the given heuristic, the search space contains exactly two alternatives for k under each instance of N_C, i. e. $k = 1$ and $k = k_{max}\left(N_C, D_T\right)$. Consequently, the size of the search space

reduces to

$$\sum_{N_C=0}^{N_{C,max}} \sum_{n=2}^{n_{1,opt}} \sum_{N_P[0]=0}^{min(n_{P,max},\, n-1-N_C)} C\left(n-1-N_P[0],\, N_C,\, 1,\, n_{P,max}\right) \tag{5.13}$$
$$+ \sum_{n=k_{max}+1}^{n_{k_{max},opt}} \sum_{N_P[0]=0}^{min(n_{P,max},\, n-k_{max}-N_C)} C\left(n-k_{max}-N_P[0],\, N_C,\, 1,\, n_{P,max}\right).$$

The complexity of the search space is further reduced by an additional heuristic derived from an inherent property of the objective function. Since reactive repair cycles of the protocol are each applied with the failure probability of the previous cycle, their probability of occurrence decays exponentially with the erasure probability (Equation 4.26). Consequently, it is a valid assumption that later cycles should transmit more repair packets than earlier cycles in order to reduce the overall expected redundancy information. Based on the above heuristic, only those parity schedules are considered that specify the portions of repair packets in ascending order towards the last repair cycle, i. e. $N_P[c] \leq N_P[d]$ holds true for any cycles c, d with $0 \leq c < d \leq N_C$. All repair packet schedules that do not meet this requirement are discarded.

The search is in the following restricted to the *integer partitions* [7] of r, that form a subset of the integer compositions in that the ordering of the single parts is arbitrary. Hence, as a convention, the parts are usually selected in ascending or descending order. $P(r)$ is defined to be the number of integer partitions of r. Similarly, let $P(r, s, a, b)$ be the number of restricted partitions of r, where s is the number of parts and a, b are minimum and maximum size of the parts, respectively [113]. Table 5.1 enumerates the $P(8, 3, 1, 5) = 4$ integer partitions of 8 with 3 parts of sizes between 1 and 5.

Finally, the reduced size of the search space is formulated as

$$\sum_{N_C=0}^{N_{C,max}} \sum_{n=2}^{n_{1,opt}} \sum_{N_P[0]=0}^{min(n_{P,max},\, n-1-N_C)} P\left(n-k-N_P[0],\, N_C,\, max(1, N_P[0]),\, n_{P,max}\right) \tag{5.14}$$
$$+ \sum_{n=k_{max}+1}^{n_{k_{max},opt}} \sum_{N_P[0]=0}^{min(n_{P,max},\, n-k_{max}-N_C)} P\left(n-k-N_P[0],\, N_C,\, max(1, N_P[0]),\, n_{P,max}\right).$$

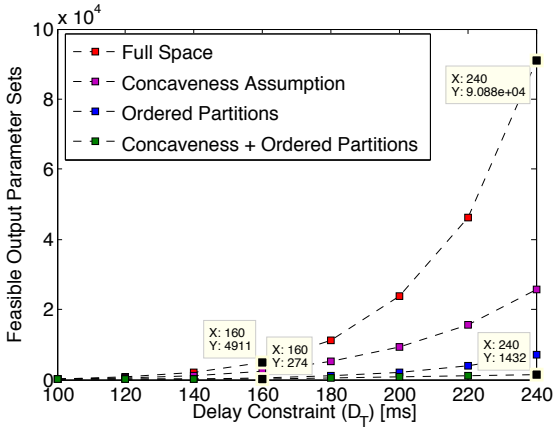
Figure 5.1 compares the size of the full and the reduced search space for the optimization of the coding parameters under a specific example. A packet stream with packet interval $T_S = 2.5\,ms$ is assumed to be transmitted on a network with packet loss rate $P_e = 0.1$ and $RTT = 20\,ms$. As the delay constraint D_T increases, the number of feasible combinations of k and N_C grows, while a larger k also requires larger repair packet schedules. As a result, the full search space grows exponentially with D_T. Under the combination of the concaveness assumption for the block length and the restriction of the repair schedules to ordered integer partitions, the space grows with a very small exponent of the delay constraint.

Greedy Search

Greedy algorithms quickly approach the optimum solution by following the objective function's gradient of highest gain. As expressed by Equation 4.26, the coding gain

Algorithm 5.1 Optimize protocol parameters.

```
1: k_opt ← 0
2: N_C,opt ← 0
3: N_P,opt ← 0
4: p_r,opt ← 1
5: ri_opt ← ∞
6: N_C,max ← ⌊(D_T − D_FEC,min)/D_REQ⌋
7: for N_C = 0 → N_C,max do
8:     k_max ← min(k(N_C, D_T), k_lim)
9:     for k = 1 → k_max do
10:        n ← ESTIMATE_N_FOR_K(k)
11:        N_P ← GEN_REPAIR_SCHEDULE (n, N_C)
12:        p_r ← P_r(k, sum(N_P))
13:        ri ← RI(k, N_C, N_P)
14:        r_eff ← R_eff(ri, R_S)
15:        IF r_eff < R_T THEN
16:            N_P ← MIN_REDUNDANCY (k, N_P, N_C, P_T, R_T, R_S, p_r, ri, r_eff, p_r,opt, ri_opt)
17:        ELSE IF p_r,opt > P_T THEN
18:            N_P ← MIN_RESIDUAL_LOSS (k, N_P, N_C, P_T, R_T, R_S, p_r, ri, r_eff, p_r,opt, ri_opt)
19:        END IF
20:        IF THEN N_P > 0
21:            p_r ← P_r(k, sum(N_P))
22:            ri ← RI(k, N_C, N_P)
23:            r_eff ← R_eff(ri, R_S)
24:            IF r_eff ≤ R_T THEN
25:                IF p_r < p_r,opt THEN
26:                    p_r,opt ← p_r
27:                    k_opt ← k
28:                    N_C,opt ← N_C
29:                    N_P,opt ← N_P
30:                    IF p_r ≤ P_T ∧ ri < ri_opt THEN
31:                        ri_opt ← ri
32:                    END IF
33:                END IF
34:            END IF
35:        END IF
36:    END FOR
37: END FOR
```



Result Space	Evaluation Complexity	
#Elements	#add.	#mul.
274	248,534	171,407
1,432	1,787,576	896,887
4,911	3,665,590	2,508,491
90,875	68,887,767	25,651,207

Figure 5.1.: Size of the result space in terms of feasible repair packet schedules under the given heuristics for different delay constraints D_T ($P_e = 0.1$, $RTT = 20\,ms$, $T_S = 2.5\,ms$)

increases gradually as the repair packet schedule is optimized. Based on the assumption that later repair cycles contribute probabilistically less redundancy, a suitable greedy search algorithm has been proposed by Tan [147] that preferably increases the repair packet schedule in the final cycle in order to diminish the residual packet loss rate with minimum increase of the redundancy information. Under the adaptive HEC architecture specified in Section 2.1.1, however, it is not necessarily optimal to concentrate repair packets on the latest repair cycles. The optimum repair packet schedule coincides with an integer partition of the available parity packets in ascending order. In addition, a particularly unequal distribution of the repair packets among the N_C cycles causes a large peak-to-average bandwidth ratio, which increases the probability of queue saturation in the network. The following search algorithm efficiently constructs and evaluates the search space under consideration of the delay constraint, the reliability constraint and the rate constraint.

A relaxation of the reliability constraint is required if no solution exists that fulfills all three constraints simultaneously. Therefore, the algorithm must decide first whether the reliability constraint or the bandwidth constraint limits the search (Algorithm 5.1). From a feasible assignment of N_C and k it first finds a good initial estimate of the code word length n. This might be achieved by iteratively evaluating Equation 4.2 with some reasonable choices for n. For instance in Algorithm 5.2 the algorithm samples feasible choices of n with a defined step size τ.

After determining n, the $n-k$ repair packets must be arranged into an efficient repair packet schedule N_P with N_C reactive cycles. N_P is obtained via Algorithm 5.3, which builds an ascending, restricted integer partition of $n-k$. The algorithm initializes each of the frontmost $N_C - 1$ reactive repair cycles with 1 and assigns the remaining repair packets to the final cycle. Afterwards, it erodes the repair packet schedules starting from the last cycle as follows. Let c and d denote two subsequent repair cycles with $0 \leq c < d \leq N_C$. As long as $N_P[c] + \zeta < N_P[d]$, the algorithm removes one repair packet from cycle d and adds it to cycle c, where ζ is an adjustable, integer slope threshold. The algorithm propagates the packet from later repair cycles to earlier cycles until the difference between subsequent cycles does not exceed the slope threshold. The erosion

Algorithm 5.2 Estimate code word length n.

Require: n_{max}
Require: τ
function ESTIMATE_N_FOR_K(k)
    $n \leftarrow k + 1$
    while $P_r(k, n) > P_T \wedge n \leq n_{max} - \tau$ **do**
        $n \leftarrow n + \tau$
    end while
    return n
end function

Algorithm 5.3 Generate repair packet schedule N_P.

Require: $n_{P,max}$
Require: ζ
function GEN_REPAIR_SCHEDULE(n, N_C)
    $N_P \leftarrow [0, ones(N_C - 1), n_{opt} - N]$
    for $n_C = N_C \rightarrow 0$ **do**
        while $N_P[n_C] + \zeta < N_P[n_C + 1] \vee N_P(n_C + 1) > n_{P,max}$ **do**
            $N_P[n_C] \leftarrow N_P[n_C] + 1$
            $N_P[n_C + 1] \leftarrow N_P[n_C + 1] - 1$
        end while
    end for
    return N_P
end function

Algorithm 5.4 Minimize redundancy information.

Require: n_{max}
function MIN_REDUNDANCY($k, N_P, N_C, P_T, R_T, R_S, p_r, ri, r_{eff}, p_{r,opt}, ri_{opt}$)
    while $p_r > P_T \wedge sum(N_P) < n_{max}$ **do**
        $(n_P, ri) \leftarrow$ INC_REPAIR_SCHEDULE(k, N_C, N_P)
        $r_{eff} \leftarrow R_{eff}(ri, R_S)$
        IF $r_{eff} > R_T$ **THEN**
            RETURN N_P
        END IF
        $N_P \leftarrow n_P$
        $p_r \leftarrow P_r(k, sum(N_P))$
        IF $p_r \geq p_{r,opt} \wedge ri \geq ri_{opt}$ **THEN**
            RETURN 0
        END IF
    END WHILE
END FUNCTION

Algorithm 5.5 Minimize residual packet loss rate.

```
1: function MIN_RESIDUAL_LOSS(k, N_P, N_C, P_T, R_T, R_S, p_r, ri, r_eff, p_r,opt, ri_opt)
2:     while r_eff > R_T ∧ sum(N_P) > k do
3:         (N_P, ri) ← DEC_REPAIR_SCHEDULE(k, N_C, N_P)
4:         r_eff ← R_eff(ri, R_S)
5:         p_r ← P_r(k, sum(N_P))
6:         IF p_r ≥ p_r,opt ∧ ri ≥ ri_opt THEN
7:             RETURN 0
8:         END IF
9:     END WHILE
10:    RETURN N_P
11: END FUNCTION
```

terminates as soon as all subsequent repair packet schedules do not differ by more than the threshold ζ in ascending order and all repair cycles are assigned no more than $n_{P,max}$ packets.

At this point the algorithm knows an initial setting of N_P for the current iteration of N_C and k that nearly satisfies the reliability constraint P_T, depending on the quality of the estimation of n. After determining whether the bandwidth constraint is satisfied (Figure 5.2), it splits into two branches. If the current parameter set does not exceed R_T, the algorithm continues incrementing the coding block length n until a sufficiently reliable configuration is reached while the bandwidth constraint is still fulfilled. However, if this set already exceeds the bandwidth constraint and does not satisfy the reliability constraint, it is obvious that no solution exists that fulfills all three constraints. In this case, the algorithm starts to decrement n.

As an outcome of the reliability model, the alteration of n changes the residual packet loss rate independently from the actual repair packet schedule N_P. However, depending on the position of the increment or the decrement within N_P, the modification has significant impact on the redundancy information and consequently the effective rate of the protocol stream. In other words, at least one repair cycle exists that adds a minimum amount of redundancy information if being incremented by one and similarly, at least one cycle exists that reduces the effective rate by a maximum amount if being decremented by one. The greedy search algorithm must determine these positions of the repair packet schedule in order to modify it with maximum gain.

Algorithm 5.6 formulates such increment and decrement operations for N_P, respectively. The increment function adds an additional repair packet to the latest repair cycle and iteratively moves it to earlier cycles while evaluating the required redundancy information. The function terminates upon experiencing increasing redundancy and returns the repair packet schedule modified at the optimum position. In contrast, the decrement function removes one repair packet from the first non-zero repair cycle, which is either the proactive repair cycle or the first reactive repair cycle. After evaluating the resulting redundancy information it either returns the modified schedule or repeatedly performs the decrement on the subsequent cycles until it experiences increasing redundancy.

Algorithm 5.6 Optimize repair packet schedule for minimum redundancy.

Require: $n_{P,max}$

```
1: function INC_REPAIR_SCHEDULE(k, N_C, N_P)
2:     ri ← ∞
3:     n_p ← 0
4:     for c = N_C → 0 do
5:         if N_P[c] ≥ n_{P,max} then
6:             continue
7:         end if
8:         if c < N_C then
9:             if N_P[c] = N_P[c + 1] then
10:                continue
11:            end if
12:        end if
13:        n_P ← N_P
14:        n_P[c] ← n_P[c] + 1
15:        ri_new ← RI(k, N_C, n_P)
16:        if ri_new ≥ ri then
17:            return (N_P, ri)
18:        else
19:            ri ← ri_new
20:        end if
21:    end for
22:    return (n_P, ri)
23: end function
```

```
1: function DEC_REPAIR_SCHEDULE(k, N_C, N_P)
2:     ri ← ∞
3:     n_p ← 0
4:     for c = 0 → N_C do
5:         if c ≤ N_C ∧ ((c = 0 ∧ N_P(c) < 1) ∨ (c > 0 ∧ N_P(c) < 1)) then
6:             continue
7:         else if c > 0 then
8:             if N_P[c] = N_P[c + 1] then
9:                 continue
10:            end if
11:        end if
12:        n_P ← N_P
13:        n_P[c] ← n_P[c] − 1
14:        ri ← RI(k, N_C, n_P)
15:        if ri_new ≥ ri then
16:            return (N_P, ri)
17:        else
18:            ri ← ri_new
19:        end if
20:    end for
21:    return (n_P, ri)
22: end function
```

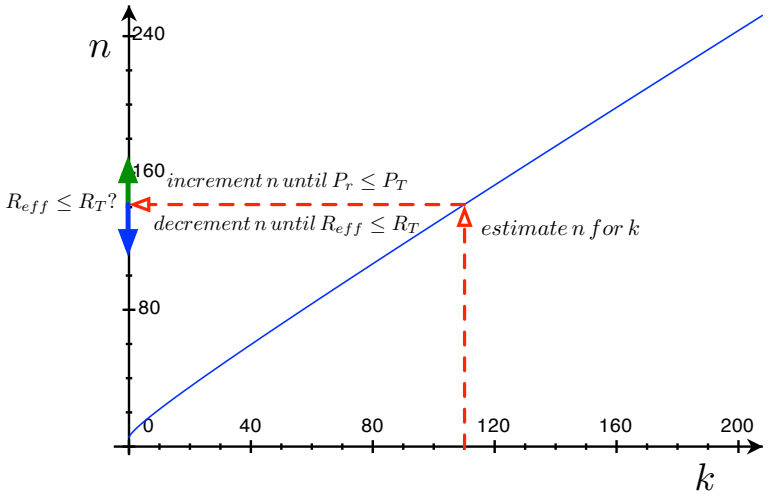


Figure 5.2.: Graphical description of the search algorithm.

Linear Prediction of the Code Word Length

The complexity of the presented greedy search algorithm depends strongly on the accuracy of the initial estimate of the code word length n. In addition, the process of finding a good estimate of the parameter by naively iterating over n is an expensive process for large k. The following algorithm exploits a known relationship between k and n in order to provide a closed-form expression for n.

As a result of Shannon's channel coding theorem, the number of source symbols and coded symbols are expected to follow a linear relationship since the code rate determines a proportional fraction of overhead symbols among the transmission symbols in the long-term average. In fact, the necessary code word length of an algebraic block-erasure code configured to achieve a specific residual erasure rate is well approximated by a linear function of k. Therefore, it is possible to predict the maximum code rate of such codes under a given reliability requirement based on a linear model.

The linear function is essentially determined by the network's packet erasure rate P_e and the desired residual packet erasure rate P_T. The function has an anchor point at $k = 1$, i. e. for the case of a repetition code. In this case the number of repair packets is predicted by the following relationship under the assumption that the erasure rate decays exponentially with the number of packet repetitions:

$$\begin{aligned} & P_T && \geq P_e^n \\ \Leftrightarrow \quad & n && = \left\lceil \frac{log\,(P_T)}{log\,(P_e)} \right\rceil - 1. \end{aligned} \tag{5.15}$$

A second point of the linear equation is determined by finding the code word length n for a large k such that

$$P_r\,(k, n) \leq P_T. \tag{5.16}$$

The two anchor points determine the equation explicitly. In particular, the slope can be determined and used to predict n for any other number of source symbols.

Figure 5.3 shows that the linear equation slightly underestimates the required code word length for short block lengths. Intuitively, this result reflects the inefficiency of

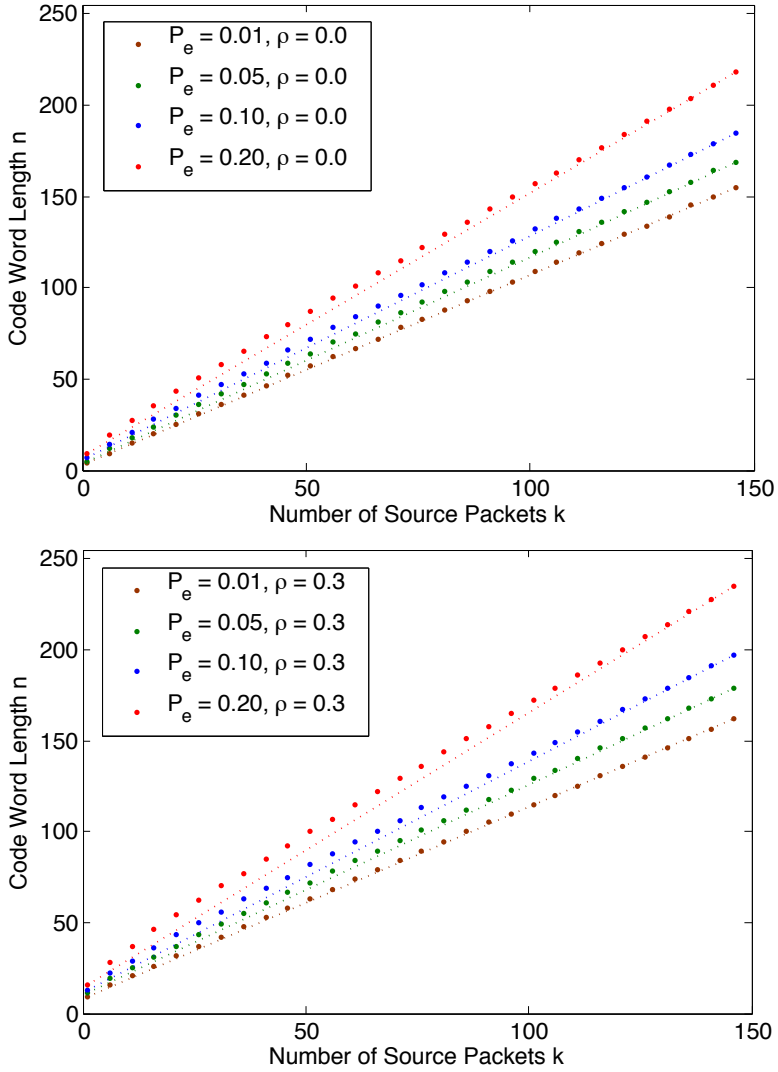


Figure 5.3.: Code word length for a given number of source packets k. Exact calculation and linear prediction (dotted line).

short channel codes. The underestimation of the required code word length is without problems as the estimate provides a good starting point for the search algorithm, which returns the exact value after a small number of increments or decrements of n. In presence of temporally correlated erasures, the linear model increasingly underestimates n for shorter code words (Figure 5.3). In this case the relationship of k and n deviates from the linear model as memory in the erasure process decreases the performance of short block-erasure codes. The underestimation of n causes few additional iterations in the search algorithm.

5.2. Adaptive Error Control

Temporal dynamics in the state of an Internet path are not predictable by static network state modeling. Packet-switching networks are considered to have a dynamic channel capacity. As a result, the configuration of the packet repair might be suboptimal in periods where the network state is better than the modeled average. On the other hand it may be insufficient to meet the application constraints in case the network state is worse than predicted by the model. In order to optimize both efficiency and performance of the transport session, an on-line adaptation of the protocol parameterization is required. Adaptive error control relies on bidirectional communication, which is a fundamental feature of IP networking.

Whereas the dynamic adaptation of the protocol is non-ambiguous in a point-to-point session, it can be arbitrarily complex in a multicast group. In particular, the protocol performance depends on the choice of a good group representative for the network state. In order to avoid the performance to be significantly degraded by one or few receivers observing severely bad network state, the thesis formulates limiting definitions for group representatives of single dimensions of the network state.

5.2.1. Reliability Control

The objective of the predictably reliable protocol is to fulfill the application's reliability requirement under a strict delay constraint as efficiently as possible. In terms of the information theory, this corresponds to an instant maximization of the code rate under the dynamic network state. The protocol follows the channel capacity dynamically. Already the reactive part of the protocol's packet repair provides a form of temporal adaptivity since it adapts the number of repair packets immediately to the number of packet erasures signaled by the receiver. However, under fixed parametrization this form of adaptivity covers a limited range of network state parameters.

The following section formulates adaptive error control as an open-loop control problem. The protocol's residual error rate is the controlled output. It is modified by the adaptation of the protocol parameters based on the observed network state under application of the protocol performance model. As a result of the open loop, the estimation of the packet loss probability must include a safety margin in order to compensate for short-term dynamics in the network path's packet loss process.

Protocol Dynamics

Predictably reliable transport relies on the close cooperation of two control mechanisms (Figure 5.4). The adaptive HEC specified in Section 2.1.1 contributes the error control functionality, i. e. it is the executive part of the architecture. Under a fixed parametrization via (k, N_C, N_P) this scheme provides partial reliability with limited delivery delay. Reliability control adds a second control circuit that configures the adaptive error control in order to efficiently strive towards a predictable residual packet loss rate. The responsiveness of the overall system to the dynamics of the network state is essentially determined by a set of update periods within these control circuits. In general, shorter update intervals enable the protocol to follow those dynamics more accurately while meeting the reliability target with higher probability at the price of increased feedback traffic and computational complexity.

The error control circuit operates at a finer granularity compared to the reliability control. It allows for parameter updates with a minimum period of one packet interval T_S. Even in case a parameter update is sent to the HEC scheme while it is about to collect the k packets for the current coding block, it immediately adopts the most recent parameter set for the next source packet by truncating the current coding block. The truncated coding block must be treated with the previous parameter set since several source packets belonging to the block might already be sent to the receiver. The truncation of the current block is required in order to ensure that the delay constraint is being satisfied for the next source packet under the updated parameter set. For the current coding block it might lead to an increased coding overhead.

The reactive part of the adaptive HEC implements short-term adaptivity to a dynamic packet loss rate. Repair packets reach the receiver upon the negative acknowledgment after a minimum latency of one RTT. However, a given protocol parameter set provides sufficient and efficient correction performance within a limited range of RTT and PLR. The long-term dynamics of the network state are therefore captured by the reliability control. The control circuit exports three parameters that remain free for specific system design.

The PLR is a statistic measure that requires a larger number of observations. The relationship between the sample size N of the observed packet sequence at the receiver and the accuracy of the block-erasure model is discussed in Section 3.2. Theoretically, the PLR can be estimated for each incoming packet with a sliding sample history of length N such that a new estimate is available with a period of T_S. Unfortunately, the PLR estimate cannot be transferred to the sender in such a high frequency as the feedback bandwidth is supposed to be limited via a minimum feedback period of T_{NFB} for network state feedback. Finally, either T_{NFB} or the update period of the reliability controller T_{UPD} define the minimum granularity of the dynamic protocol configuration. Specific settings for the free system parameters are provided along with the experimental evaluation of the protocol in Sections 6.2 and 6.3.

Controller Architecture

Figure 5.5 presents the system architecture of the reliability controller. The controller samples the network state and predicts protocol parameters by fitting the protocol's timing and block-erasure models and evaluating the protocol performance model. Based on both models the controller optimizes the protocol parameters. The protocol is the actuator of the controller and is updated with the output of the reliability controller.

The control theory differentiates between open-loop and closed-loop circuits as the two fundamental controller architectures [8]. Open-loop control requires a-priori knowledge about the network path. Therefore, the system output of the reliability control relies strongly on the accuracy of the block-erasure model and the protocol performance model. Closed-loop reliability control relies on feedback about the system output, which is the residual packet erasure rate in case of the reliability controller. A closed circuit detects a potential prediction error of the models by comparing the actual residual erasure rate with the reliability constraint of the application, which is the set point of the controller. Moreover, the closed loop can compensate for the influence of system perturbations and temporal dynamics of the network path, i. e. error sources that are not covered by the applied models.

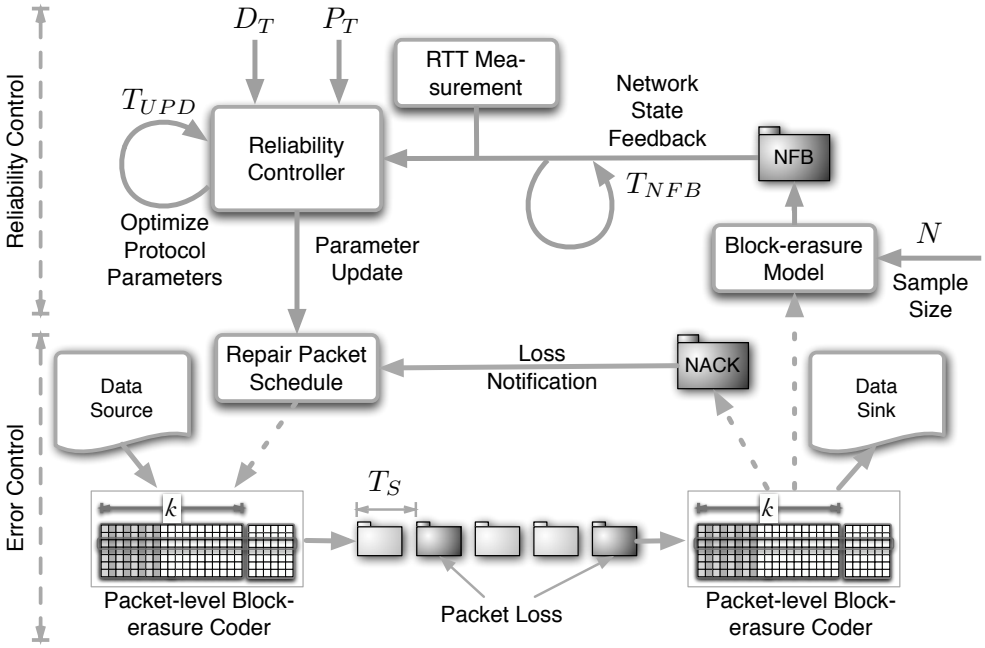


Figure 5.4.: Protocol dynamics.

Closed-loop control causes several problems in the given application scenario of reliability control. First, especially for tight reliability constraints the sensing of the system output is difficult since the transmission of a large number of packets must be analyzed in order to obtain statistically accurate feedback about the residual erasure rate. Second, a feedback loop may introduce large convergence delays, whereas the open-loop controller immediately provides a stable output. Third, as the samples of the network state must traverse the return network path, the controller receives feedback at least one RTT after producing the corresponding output. The feedback delay has additional impact on the convergence delay of the control circuit. In order to bypass these problems, the reliability control of the predictably reliable protocol is specified as an open-loop controller. Hence, protocol parameters are adjusted purely based on the predictions of the timing model, the block-erasure model and the protocol performance model. Feedback on the residual packet loss rate that is experienced by the receiver is not considered in the controller.

Margin of Error

In response to random perturbations in the system, the reliability controller might suggest protocol parameters that are either too weak to meet the reliability constraint or it causes over-provisioning of redundancy information. Whereas the former obviously violates application constraints, the latter just results in slightly suboptimal bandwidth utilization. Therefore, the controller can compensate for potential inaccuracies of the model by a certain over-provisioning in the estimation of protocol parameters, without affecting the primary objective of fulfilling the reliability constraint.

The accuracy of the block-erasure model is formulated in Section 3.2 in dependence on the number of measurement samples and a chosen confidence level. Under the dynamic network state, a smaller history of measurement samples is preferred in order to

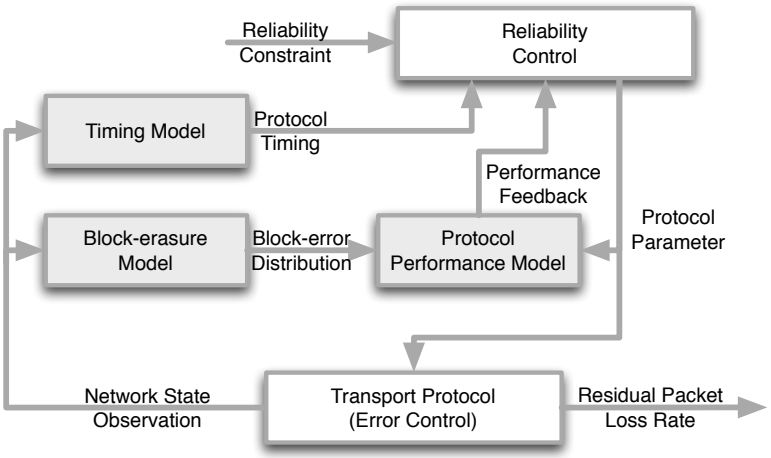


Figure 5.5.: Reliability controller.

improve the responsiveness of the controller. The accuracy of the block-erasure model is given as a percentage of the estimated mean values (Equation 3.22 and Equation 3.38), which defines a margin of error around the estimates. Consequently, the controller can compensate for the model's deviation from the actual packet loss process by adding the margin of error for the given confidence level to the mean estimate.

As the estimator becomes more accurate with increasing sample size and vice versa, the margin of error can implicitly be controlled via the choice of the sample size. Figure 5.6 shows a measured trace of the packet loss rate on a wireless home network. The average PLR is estimated via the Bernoulli model with a sliding window of size $N = 1024$ (left) and $N = 512$ (right), respectively, on the measurement samples. The margin of error is calculated for a 99 % confidence interval. The average PLR has a larger amplitude as a result of the shorter scope in case 512 instead of 1024 samples are evaluated. In addition, the margin of error increases significantly as a result of the limited sample size. In order to obtain conservative protocol parameters, the block-erasure model represents the compensated PLR estimate, i. e. the mean PLR increased by the margin of error. The experimental results in Chapter 6 are obtained based on the estimation of the respective block-erasure model from a sample size of $N = 512$ with a 99 % confidence interval.

Update Period

The update period T_{UPD} implicitly determines the progression of the sample window during the network state observation. Hence, it also affects the responsiveness of the protocol to variations in the network state. In the architecture of the predictably reliable protocol, this parameter is left for manual adjustment. However, the following discussion identifies two observations that justify a reasonable choice for the update period.

First, it may reflect the network path's coherence time, i. e. the period in which the network state is statistically well predictable via an instance of the protocol's timing and block-erasure models. The coherence time depends strongly on the physical properties of the network infrastructure. For instance, wireless and mobile paths cannot be assumed to be coherent beyond a period of few hundred milliseconds as interfering signals and obstructions might degrade the signal reception at any time. On the other hand, wired

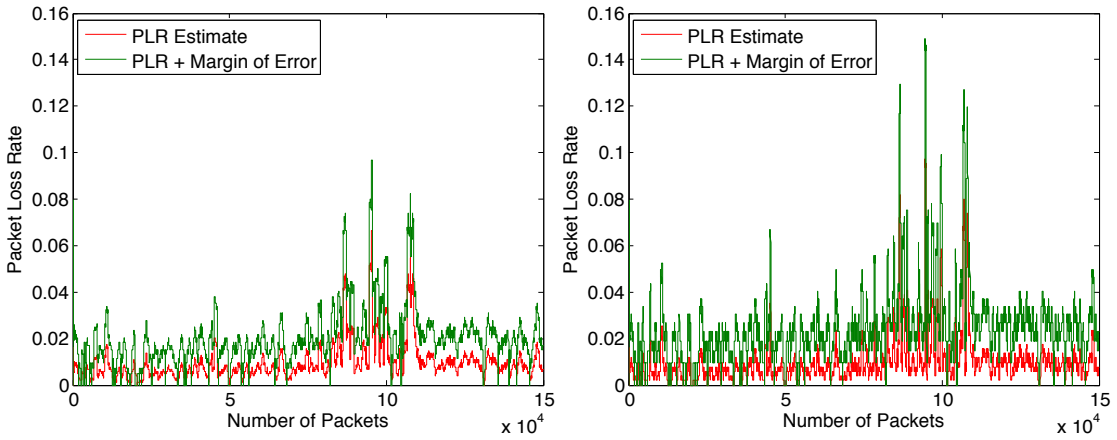


Figure 5.6.: PLR estimate; sample size 1024 (left) and 512 (right), margin of error added for 99% confidence level.

Internet paths are physically stable but they experience fairly periodic congestion events due to TCP's congestion control policy. The period depends mainly on the number of TCP flows transmitted in parallel as well as the path's RTT (see further Section 6.2.1) and may be in the order of several hundred milliseconds or up to few seconds.

A second limiting factor for the update period of the control system derives jointly from the complexity of the optimization problem and the available computing capacity. As evidenced by Figure 5.1, the complexity of the search algorithm is in the order of 10^6 to 10^7 floating point operations. Current personal computer systems perform in the order of 10 to 100 GFLOPs[2]. Therefore, the computing capacity of such platforms does not dictate a reasonable limitation on the update period since parameter updates within an interval of few hundred milliseconds are possible in order to address short coherence times of the network infrastructure. The experiments presented in Chapter 6 successfully apply an update period of 200 ms within wired as well as wireless Internet scenarios. In general, the best protocol performance is being achieved if the application's delay constraint is sufficiently greater than the RTT to allow for at least one reactive repair cycle such that short-term dynamics of the packet loss process can be captured by the error control.

Dynamic Code Rate

As a result of the reliability control, the predictably reliable protocol optimally follows the network path's dynamic channel capacity. Other than a fixed FEC scheme that deterministically adds a specific number of parity packets to the source packets in order to achieve the desired code rate, the protocol stochastically shares the available bandwidth between source and repair packets, depending on the dynamic network state. The protocol's efficiency model calculates the expected amount of repair data that is added to the source stream under a certain protocol configuration (k, N_C, N_P) (Section 4.3.1). Under the optimization problem from Section 5.1.1 the reliability control results in a *dynamic code rate*. The dynamic amount of redundancy information RI_{min} is adap-

[2] 10^9 floating-point operations per second.

tively approached by the incremental redundancy scheme and the periodic parameter optimization (Equation 5.1). Based on RI_{min}, the dynamic code rate R_d is defined as

$$R_d = \frac{1}{1 + RI_{min}(k, N_C, N_P)}. \tag{5.17}$$

5.2.2. Multicast Parameter Optimization

It is a major problem in adaptive multicast error control to find good representatives for both packet loss rate and round trip delay within the multicast group. The literature defines the *currently limiting receiver* (CLR) as the receiver that limits the protocol's efficiency because it is currently experiencing the worst network conditions among the receiver group in an essential observation criterion. The term has been introduced in Widmer's work on multicast congestion control [163, 162], where the CLR is the receiver whose conditions would lead to the lowest throughput.

In a wide area network it is likely that a small fraction of the receiver group experiences a significantly larger loss rate or delay compared to the remainder of the group. This could be the result of congestion in a specific network segment shared by a small subset of the multicast group. Similarly, one or few wireless receivers might experience strong interference or weak signal such that they suffer from increased physical packet loss. In those situations a single receiver with bad reception quality dictates the protocol performance and the efficiency for the entire group.

This section introduces the *group round trip time* (GRTT) [163, 3], *group packet loss rate* (GPLR) as well as the *group correlation coefficient* (GCC) as the three essential parameters to represent the multicast network state. Both parameters represent either the worst case conditions within the entire multicast group or within a tunable percentage of the group.

Multicast Delay and Packet Loss

The CLR represents the dimensions of the network state, which include RTT, PLR as well as the correlation coefficient CC ($= \rho$) if it is determined by the block-erasure model. The network state determines the protocol goodput indirectly as a result of the protocol parameter optimization. Therefore, this thesis defines the CLR to be the receiver that causes the minimum protocol goodput under a given bandwidth constraint. This in turn corresponds to the maximum redundancy requirement or equivalently the lowest dynamic code rate among the multicast group.

A major problem of the CLR concept comes up if the CLR's network state represents a small minority of the receiver group such that the service for most of the group members is degraded despite their good network conditions. Reliable multicast does not address this problem since even the worst receiver must experience total reliability, independently of how much it differs from the majority of the group. However, in case of partial or predictable reliability it is feasible to aim at an overall service maximization within the whole receiver group. Effectively, the multicast of loss-tolerant content may imitate the limited coverage area in the digital media broadcast, where a receiver must reside in a range sufficiently close to the broadcast antenna in order to experience a good reception quality.

Predictably reliable transport provides means to limit the reception range via the reliability control. The coverage of the predictably reliable protocol is implicitly controlled by the choice of group estimates for RTT, PLR and CC. The sender requires knowledge about the RTT in order to calculate suitable protocol timers as defined in Section 2.2.2, whereas PLR and CC determine the coding parameters via the block-erasure model's block error distribution. Therefore, the incoming network state samples obtained from the receiver feedback need to be consolidated into suitable group representatives:

- **Group Round Trip Time (GRTT):** The GRTT is the basis for the estimation of protocol timers that are valid for the whole receiver group in order to finalize transport and error control under the delay constraint.
- **Group Packet Loss Rate (GPLR)** and **Group Correlation Coefficient (GCC):** GPLR and GCC influence the parametrization of the block-erasure code, in particular, the number of repair packets to be sent proactively and reactively in order to satisfy the reliability constraint and the delay constraint.

Receiver Group Size

Knowledge of the *receiver group size* N_R is essential for the optimization of the protocol parameters in the multicast. It is a variable of the feedback model and the efficiency model. For a larger group size the feedback model leads the reliability control to increase the timer interval for the feedback suppression timer and the period of the network state feedback so as to fulfill a given constraint on the feedback bandwidth. At the same time the efficiency model pronounces the advantage of longer coding blocks for increasing N_R.

Friedman and Towsley apply probabilistic polling in order to estimate the multicast session size [61]. The sender sends out polling requests within r rounds on which each receiver answers with an i. i. d. probability. The number of answers is thus binomially distributed such that the problem of counting the receivers translates into estimating parameter q for the binomial distribution $binomial(q, r)$ while r and the success probability are known. If protocols perform bookkeeping on the session state, such as implemented in RTP/RTCP [135], an exact count of the session size is available. Similarly to RTP/RTCP, PRRT specifies periodic network state feedback. Network state feedback is sent with the interval T_{NFB} by each receiver, whereas the interval is chosen sufficiently large in order to preserve the scalability. After several feedback periods, each receiver is expected to have successfully communicated with the sender.

The receiver bookkeeping functionality is most efficiently implemented via hashing. Hashing enables insert and lookup operations in amortized constant time complexity [42]. A lookup operation on the receiver identification is performed as soon as a feedback message reaches the sender. The receiver is added to the bookkeeping data structure if the lookup returns without result. The number of elements in the hash table is updated upon each insert or delete operation. A receiver must be deleted from the bookkeeping structure if it stays inactive for a specific period of time. Since the network state feedback is sent periodically, a particular receiver should be removed if the sender does not observe any feedback from this receiver after a small multiple of T_{NFB}. This requires a periodic search through the entire hash table, which causes linear complexity in the measured number of receivers. An over-estimation of the receiver group size is not critical for the protocol's reliability as N_R is not a parameter of the reliability model. It might, however,

temporally affect the transport efficiency. Hence, the expiration of receivers should be checked with a large period in the order of few seconds in order to preserve the scalability of the receiver bookkeeping.

Limited Coverage

Commonly, the GRTT is defined in the literature as the maximum RTT among the multicast group. However, under this definition one or few statistical outliers can have significant impact on the protocol performance. In the following the definition of the GRTT is proposed as the maximum RTT within a certain percentile of the receiver group such that the GRTT is a representative for a tunable fraction of the receiver group.

Assume the network state samples to be available in decreasing order. Let ϕ be the desired percentile of receivers covered by the CLR's network state estimate. Then, ϕ determines the rank s of the network state sample that is greater or equal than the sample of ϕ percent of the receiver group with size N_R:

$$s = \left\lfloor \left(1 - \frac{\phi}{100}\right) \cdot N_R \right\rfloor . \tag{5.18}$$

If the coverage of the entire receiver group is desired, the equation evaluates to $s = 0$ and the worst network state sample is chosen to represent the group. As predictable reliability should be served to the majority of the group, it is valid to assume that $s \ll N_R$. Let the sender maintain a set of $s+1$ RTT samples $\{RTT_0, \ldots, RTT_s | 0 \leq i \leq j \leq s \Rightarrow RTT_i \geq RTT_j\}$ in decreasing order from $s+1$ different receivers of the multicast group. The GRTT is chosen as $GRTT = RTT_s$. Receivers with $RTT > GRTT$ perceive a higher residual erasure rate since they might suffer from the late reception of repair packets. Their aggregate codeword length after consumption of the time budget D_T is reduced with high probability due to incomplete or missing repair cycles.

Similarly, the GPLR is defined as the maximum PLR among a percentile of the receiver group. Let $\{PLR_0, \ldots, PLR_s | 0 \leq i \leq j \leq s \Rightarrow PLR_i \geq PLR_j\}$ be a set of $s+1$ PLR samples from $s+1$ different receivers in decreasing order. Let s be the rank of the PLR sample representing ϕ percent of the receiver group as defined in Equation 5.18. Then the GPLR is set to $GPLR = PLR_s$. Finally, let $\{CC_0, \ldots, CC_s | 0 \leq i \leq j \leq s \Rightarrow CC_i \geq CC_j\}$ be a set of $s+1$ samples of the CC from $s+1$ different receivers in decreasing order. Let s be the rank of the sample representing ϕ percent of the receiver group as defined in Equation 5.18. Then the GCC is set to $GCC = CC_s$.

The estimation of the group representatives based on the limited coverage requires the sender to maintain a list of the $s+1$ worst samples of each dimension of the network state among a group of N_R receivers. Let N_R be estimated via the above receiver bookkeeping. Further, let T_{NFB} be derived depending on N_R in order to limit the bandwidth consumed by the network state feedback. For each network state feedback that arrives at the sender, the sender compares the incoming samples of RTT, PLR and CC with the $s+1$ worst samples stored in the corresponding list. If any of the incoming samples belongs to the $s+1$ worst network state observations in one of these three measures, the respective dimension of the network state is updated. In this case the new sample is inserted into the list, whereas the sample with former rank s is deleted. A list entry expires after an adjustable multiple of T_{NFB}. This functionality is efficiently implemented by a heap data structure while causing time complexity $O(s \log s)$ and space complexity $O(s)$ [42].

Receiver Admission

In order to execute the limited coverage with respect to the network state, the sender must suitably react on loss notifications from receivers that are not within the coverage region. Otherwise, those receivers might implicitly become the CLR as they activate excessive sending of reactive repair packets, which reduces the goodput for the entire multicast group. Due to the fact that the network state information is carried within any feedback packet (Section 2.1.3), the sender can reject loss notifications carrying network state information that are worse than the group representatives in at least one dimension. This is possible for PLR and CC by explicitly comparing the fields with the respective group representative. For a receiver admission based on the RTT, the sender must first calculate this receiver's RTT based on the **RTT probe** field in the feedback packet. If the resulting RTT is greater than the GRTT, the transmission of the corresponding repair packets is omitted unless a loss notification from an admitted receiver requests repair information for the same coding block. Even though loss notifications from receivers without admission are ignored, their updates on the network state must always be considered in the periodic calculation of GRTT, GPLR and GCC.

In case a multicast receiver experiences an excessively bad network state for a certain period of time, it is temporally turned into a passive participant of the protocol session by execution of the receiver admission scheme. Under the limited coverage of predictable reliability, the receiver still benefits from the repair packets requested by admitted group members, without degrading the overall protocol performance. The admission scheme is updated with the period of the network state feedback.

5.3. Congestion Control

Congestion control is the approach to implicitly *share a limited bandwidth* among several network flows of the same type or competing flows of other protocols while maintaining *long-term fairness* in the rate allocation. It relies on the assumption that each network flow probes the network bandwidth and actively reduces its sending rate upon sensing congestion. The equal share of the bandwidth is obtained via a distributed control algorithm. Control algorithms are supposed to react quickly to temporal variations of the available bandwidth e. g. due to multiple access or dynamic channel capacity. Congestion control is the Internet's prevalent multiple access scheme caring for a stable coexistence of a vast number of competing protocol flows.

Predictably reliable error control requires a recent estimate of the dynamic network bandwidth that is available to the protocol flow. In case of a shared network, this value is obtained from a congestion control mechanism. Especially the combination of error control and congestion control remains challenging since packet loss affects the congestion signal of most control algorithms, whereas error control increases the bandwidth requirement. The protocol performance model of the predictably reliable protocol is able to stochastically predict the bandwidth requirement of the error control. Under an estimate of the available bandwidth, it explicitly calculates the temporary protocol goodput. This goodput is achieved with the desired predictable reliability. The goodput estimate together with the expected level of reliability is signaled to the application on top of the protocol layer.

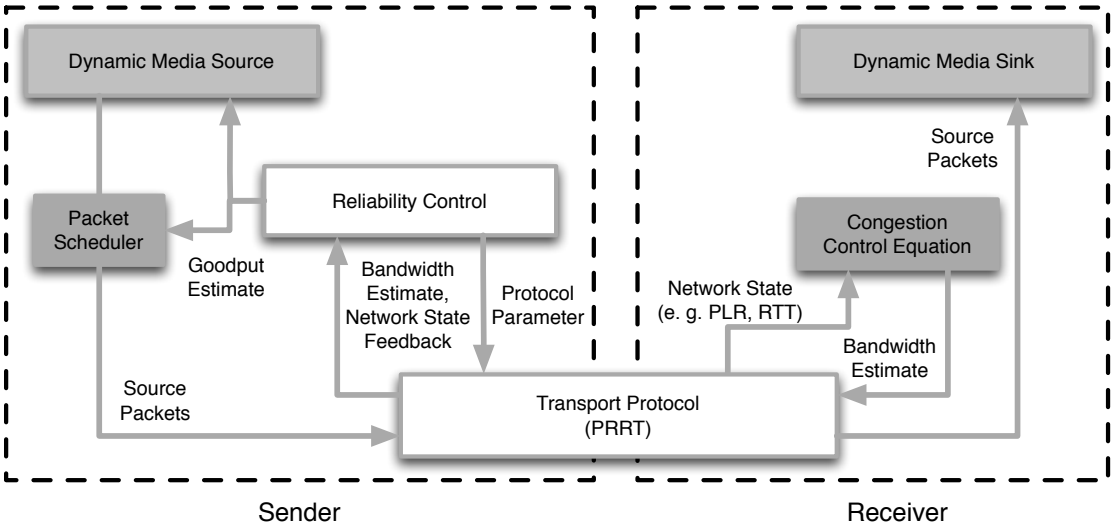


Figure 5.7.: Congestion controller with dynamic media source.

5.3.1. Controller Architecture

Since the error control framework requires an estimate of the available link capacity, the protocol implements a rate control framework that relies on *equation-based* congestion control. In contrast to *window-based* control schemes – such as applied in TCP – that just perform implicit probing of the maximum throughput via an AIMD algorithm, equation-based congestion control returns an explicit estimate of the bandwidth share. Importantly, window-based congestion control is not feasible for multicast applications as it requires positive acknowledgments, which cause feedback implosions in large receiver groups.

The following rate control framework (Figure 5.7) puts all information into place in order to support different loss- and delay-based control equations. The rate controller implements a simple feedback loop, where the output of the control equation is the maximum throughput rate of the protocol flow. The throughput estimate of the control equation is applied to a packet scheduler that acts as the control device and ensures the maximum sending rate for the controlled network flow. The packet scheduling is required as the protocol does not implement positive acknowledgments in order to clock the sending rate. The scheduler operates according to a defined prioritization policy that is optionally refined by the application.

Control Equation

The literature understands congestion control as an optimization algorithm that attempts to solve the problem of network utilization maximization [168, 31] in a shared network. The general formulation of a NUM problem comprises a set of network sources, a set of links in a meshed network and a routing matrix that defines the link usages by each packet flow. Sources inject packet flows at a specific rate. Their rate is managed by the congestion control while the rate controller receives feedback from the network path in form of a *price* for each rate unit, which is implicitly determined by the resources on the network path. This price may represent the packet loss rate, the frequency of packet loss

events or the queueing delay. Upon receiving the path feedback, the control equation adjusts the rate estimate by trading off price and utility as a function of the rate under the constraint of the path's overall capacity [31].

For instance, this principle has been implemented via loss-based congestion control into the TFRC protocol [162]. The protocol applies the TCP response equation from Padhye's model for TCP throughput [116] to current observations of the network state in order to estimate the throughput a TCP flow would acquire under the same conditions. Specifically it returns the steady-state throughput of TCP-NewReno [59] given the RTT and the recent loss event rate.

It has been observed that delay-based congestion control is in particular superior to loss-based algorithms under a large bandwidth-delay product. Recently developed control equations for high-speed, reliable data transfer either purely rely on delay-based congestion notifications [83, 160] or on hybrid delay-based and loss-based [4, 148, 97, 68, 25]. Those equations evaluate the ratio between the most recent RTT sample and the network's *base RTT*, i. e. the round trip delay of the unstressed network path.

Due to the fact that congestion control equations translate network state observations into a current estimate of the available bandwidth, their interface to the remaining protocol components relies simply on the exchange of those parameters. Since the predictably reliable protocol measures the relevant network state parameters and implements suitable feedback mechanisms, its congestion control policy can conveniently be modified by exchanging the applied control equation. In particular, Section 5.3.2 considers the application of a delay-based control equation and discusses its feasibility with respect to media-friendliness.

Feedback Acquisition

Equation-based congestion control makes the location of the controller arbitrary and removes the dependency on the network state in the return path. This is in contrast to window-based control, where the positive acknowledgment from the receiver is required at the sender to move the congestion window forward on the outgoing packet queue. The predictably reliable protocol implements the control equation at the receiver because of two reasons. First, locating the equation at the receiver improves the scalability of the protocol and enables multicast congestion control. The receiver calculates an estimate of the feasible sending rate upon the measurement of the relevant network parameters and communicates the current rate estimate to the sender. In a multicast environment this method relieves the sender from evaluating the control equation for each single receiver. It must only implement a decision policy that chooses a feasible sending rate among the incoming receiver estimates. At the same time, this enables lightweight communication between receiver and sender, which leads to the second reason for the above decision: The control equation at the receiver compresses the different dimensions of the measured network state into a single value, i. e. the rate estimate. Therefore, this scheme is significantly more robust against the loss or the delay of feedback and it allows the interval of the feedback to be chosen independently.

Depending on the type of the control equation, different information is required about the network path in order to generate the control feedback. The loss-based algorithms rely on the measurement of the average packet loss rate as well as the loss event rate. The latter is usually replaced by the inverse of the average loss interval, i. e. the number

of successful receptions between two adjacent packet losses [162]. Both measures are conveniently derived from the loss indicator sequence maintained at each receiver (Section 2.4.2). The delay-based equation incorporates an estimate of the base RTT versus the current actual RTT of the network path. The latter is available at the receiver from the protocol's periodic RTT estimation. The literature defines the base RTT as the minimum RTT sample measured over the entire protocol session [91], which is determined at the receiver based on the retrieved samples of the current RTT. Obviously, congestion control requires periodic feedback from the receiver that is independent from the occurrence of packet loss events. The bandwidth estimate is therefore transmitted via the periodic network state feedback.

Packet Scheduling

Window-based congestion control is considered to be self-clocking due to the positive acknowledgment of each successfully delivered packet. In such a scheme the acknowledgments are serving as a clock for the window update and the injection of new segments into the network. As a result, the window progression is subject to the delay variations in both the forward and the feedback channel. This leads to severe delay jitter in the delivery of single transmission segments, which is infeasible for inelastic network flows.

In case inelastic traffic is sent under the absence of an ACK-clocked window, a control device is required that masks excessive traffic if the effective sending rate of the transport protocol exceeds the rate estimate of the control equation. A flow that exceeds the rate estimate either affects the fairness towards other protocol flows or even the network's stability by overwhelming the network path with the offered rate. Combined with the protocol's outgoing packet queue under limited buffer space, this system can be configured to reject a certain fraction of packets. Note that packets only need to be rejected under the condition that the application does not reduce the sending rate upon the protocol's notification about insufficient bandwidth. Optionally the prioritization follows a specific policy, for instance, by rejecting least significant packets first. By default, the predictably reliable protocol applies the following prioritization policy:

- **Source packets** are delivered with highest priority.
- **Reactive repair packets** of later repair cycles have second rank since their delivery increases the chance of completely recovering coding blocks that have already experienced transmissions of repair packets. The priority of reactive repair packets decreases along with the repair cycle they are scheduled for.
- **Proactive repair packets** have lowest priority.

Deviating from the default policy, the application might define a customized scheduling policy by setting the priority field in PRRT's general packet header (Section 2.1.3). This is particularly useful in combination with scalable source coding, where packets of higher enhancement layers have less significance for the reconstruction of the source signal.

5.3.2. Media-friendly Congestion Control

Other than for elastic network traffic, reliability for inelastic traffic results in variable bandwidth requirements as the extra bandwidth caused by the repair packets adds to

their rigid source rate. At the same time, several flows of both characteristics might share a common bottleneck link. Under such circumstances packet erasures result from queue saturation with high probability. Elastic flows can easily throttle down their throughput in order to drain the saturated queue. Their congestion control policy tends to equally share the available bandwidth. Media streams however might be less responsive and they suffer from quality reduction under congestion control. Rate control for media applications is a topic of ongoing research and the available thesis cannot contribute a final solution. The following section, however, discusses the state of the art of the research in this field while pronouncing the advantages of delay-based congestion control for the real-time media transport. A potential instance of media streaming under delay-based congestion control is being experimentally evaluated in Section 6.2.

Media-friendliness vs. TCP-friendliness

Corresponding to resource reservation and admission control in managed Internet, congestion control is the basic scheme for the coordination of multiple access to a shared IP network. Despite its infeasibility for a significant amount of today's Internet traffic, TCP defines the quality metric for congestion control. *TCP-friendliness* has historically evolved to a fundamental design goal for rate control in the *unmanaged Internet*. It applies to those network flows that acquire approximately the same rate as a TCP flow under similar network conditions, averaged over few round trip times [56, 172]. TCP-friendliness is mostly expressed via *Jain's fairness index* [82].

The feasibility of TCP's congestion control for high-rate, interactive multimedia applications that share the requirement of a tighter delay bound for the packet delivery has commonly been put into question [36, 32, 172, 58]. However, among the proposed modifications to TCP and the newly developed equation-based congestion control schemes a satisfying solution for media-friendly packet delivery is still missing. On the other hand, due to the vast amount of multimedia traffic existent in the world wide web, there is an immanent danger of congestion collapse or starvation of parallel TCP traffic if media applications do not implement rate control outside of managed network paths [76].

The question whether the large amount of multimedia traffic that is meanwhile delivered over unmanaged Internet postulates a redefinition of the fairness term – especially considering the trade-off between fairness and conservation of real-time requirements – has driven much research in the field of congestion control [140]. The proposed ideas include modified control equations that provide smoother response to a variable network state [141, 56, 32] or the suggestion to formulate the control equation more aggressively so as to treat multimedia flows with preference [33].

Due to TCP's serious inefficiency on wireless and mobile networks as well as network paths with a large bandwidth-delay product, *opportunistic TCP-friendliness* has been proposed as an alternative metric [156]. This fairness metric allows competing flows to acquire more than the TCP-friendly share of bandwidth on such network paths as long as the throughput of a parallel TCP sessions is not negatively affected. This definition enables more sophisticated congestion control schemes to additionally allocate the bandwidth that remains unutilized by TCP.

It is obvious that congestion control is not possible without adaptation of the inelastic media stream's source rate. Therefore, the primary goal is to maximize the source rate under congestion control. This is for instance the objective of dynamic media streaming

approaches that offer video streams in multiple bit rates, whereas a particular version is chosen depending on the estimate of the available bandwidth [74, 144]. Consequent requirements of such media streaming schemes are, however, a stable bandwidth estimate and low packet jitter.

Delay-based Congestion Control

As a result of loss-based congestion control, the queueing delay at the bottleneck builds up until packets are finally rejected at the queue. However, it was found that the load-throughput curve of a rate-controlled packet stream has a characteristic knee form, reflecting the fact that an offered rate beyond a certain point leads to a marginal throughput gain. Based on this observation, the queueing delay has been proposed as a congestion signal [81]. Delay-based congestion control is able to maintain the queue level in order to approach – but not exceed – the knee point, which maximizes the throughput under the avoidance of packet loss.

In particular, delay-based control schemes maintain a stable queue level over a longer time such that they increase the throughput and reduce the delay jitter of single transmission units. Both are desired properties of media-friendly congestion control. The delay-based control equation additionally introduces several advantages for high-rate media transmission:

- **Smooth rate control:** Delay-based congestion control determines the level of network saturation continuously from the ratio between current and minimal round trip time. This allows the equation to find an equilibrium of queue saturation in the network, which results in significantly smoother rate control compared to window-based AIMD congestion control.

- **Loss differentiation:** Loss-based congestion control is prone to physical packet loss, which is prevalent in wireless networks. It erroneously interprets all packet erasures as congestion events. As a result, most TCP flavors significantly underutilize the available network bandwidth in presence of physical packet corruption [70]. Since delay-based congestion control evaluates the queueing delay in order to obtain a congestion signal, it is not affected by the packet loss.

- **Explicit bandwidth estimation:** The control equation returns an explicit estimation of the available bandwidth. Given this estimate, the maximum goodput of the predictably reliable protocol under the current network conditions is predicted via the protocol performance model.

Major criticism against delay-based congestion control derives from the fact that the correlation between the RTT and the congestion loss has been observed to be low in earlier research work [18, 30]. Especially under a high level of flow multiplexing at a network bottleneck, the sampling of the queue level might become sparse for each individual flow [103]. This fact renders queueing delay an unreliable congestion signal for low-rate traffic. Therefore, delay-based congestion control was first supposed to be applied on network paths with a large bandwidth-delay product, where the packet frequency is particularly high [160]. Since high-quality multimedia applications offer a continuously high packet rate to the network, this condition is fulfilled. Under the large packet frequency, the delay-based control equation obtains RTT samples in a sufficiently high density such

that it can perform extensive low pass filtering in order to obtain a smooth estimate of the queueing delay.

6. Experimental Validation

"It is the weight, not number of experiments that is to be regarded."
Isaac Newton

Traditional transport protocols have been designed for a special purpose such that they rely on strong assumptions with respect to the underlying Internet infrastructure. Consequently, their performance is limited as soon as they are being applied on a different or heterogeneous physical infrastructure. As an example, TCP significantly loses throughput if it is applied on wireless or mobile network segments. Hence, more flexibility in the protocol design is desirable under the variety of today's application scenarios.

The actual idea of providing a multi-purpose protocol stack is not new in the field of digital transmission. For instance, software defined radio implements this paradigm on physical layer, where the tuner interface allows to be individually programmed for various modulation schemes and radio protocols [49]. The predictably reliable protocol provides this flexibility on transport layer. For any application scenario the protocol operates with the optimal mixture of proactive and reactive error control, which are both fundamental error control techniques on packet level. Therefore, the protocol is a basis for virtually any datagram-based error control scheme.

This chapter evaluates the protocol design and the related modeling presented throughout this thesis under the distribution of real-time video over both wired and mobile wide-area networks as well as wireless home networks. Each infrastructure exposes the transport protocol to characteristic network state parameters. Video communications services, on the other hand, have high reliability requirements and offer a rate in the order of several megabits per second to the delivery network. Video streaming applications therefore define a challenging evaluation scenario for transport layer protocols.

After a general discussion of the impact of single dimensions of the network state on protocol parameterization and performance, practical experiments are contributed for the wired as well as the wireless scenario. For both environments, the chapter measures and analyzes the characteristics of the packet erasure process. In particular, the wireless distribution is evaluated in the multicast, which requires the handling of scalability and reliability issues of the receiver feedback.

6.1. Performance Evaluation

The network state in real environments is dynamic and the protocol adapts the parametrization instantly via the optimization algorithm in Section 5.1.2. Therefore, the following section evaluates PRRT's performance theoretically under the assumption of a time-invariant environment. It evaluates the protocol's response to particular instances of the network state and points out the impact of single dimensions, such as RTT, PLR and temporal correlation. The optimal configurations as well as the achieved reliability and efficiency are predicted via the stochastic protocol performance model from Chapter 4. The theoretical performance is compared to state-of-the-art transport protocols implementing total and partial reliability. The comparison unveils in particular the operation principle of the adaptive HEC in contrast to purely ARQ-based packet repair. Finally the trade-off between target reliability and protocol goodput is formulated.

6.1.1. Response to the Network State

RTT, packet erasure probability and temporal correlation are essential parameters characterizing the network state. They appear in various combinations on different network infrastructures. Other than traditional transport protocols, PRRT compensates for the increase of both parameters with an increasing amount of redundancy.

Comparison with TCP and PR-SCTP

In order to clarify PRRT's operational mode, its behavior is compared with TCP and PR-SCTP as representatives for totally and partially reliable protocols. Figure 6.1 is an extract from a transmission via any of those protocols on a network with 10% packet loss rate. Whereas TCP translates packet loss into unbounded delivery delay, PR-SCTP leaves a large amount of residual packet losses because the purely reactive error repair cannot control the residual loss rate under a strict delay constraint. PRRT, however, maintains a constant delivery delay and adjusts the sending of repair packets in order to meet the reliability requirement defined by the application. As evident from Figure 6.1, each packet experiences constant delivery delay since, by allocating a sufficiently large end-to-end delay budget, the protocol makes the underlying network dynamics transparent to the application.

PRRT's variable is the redundancy added to the transport of the source data, depending on the proactivity of the error control. In contrast to PRRT, PR-SCTP cannot increase the proactivity of the repair process, i. e. it is neither able to send repair packets in advance nor it can send more than one packet in each reactive repair cycle. As a result, the redundancy of PR-SCTP is widely constant and drops slightly below the theoretical minimum as the RTT increases and later cycles are omitted due to the delivery deadline (Figure 6.2, left). PRRT adds more redundancy along with the increasing RTT. Hence, it keeps the delivery delay as well as the residual packet loss rate constant, despite the increasing communication delay (Figure 6.2, right). PR-SCTP, on the other hand, exposes the application to a larger residual packet loss rate due to the omitted repair cycles.

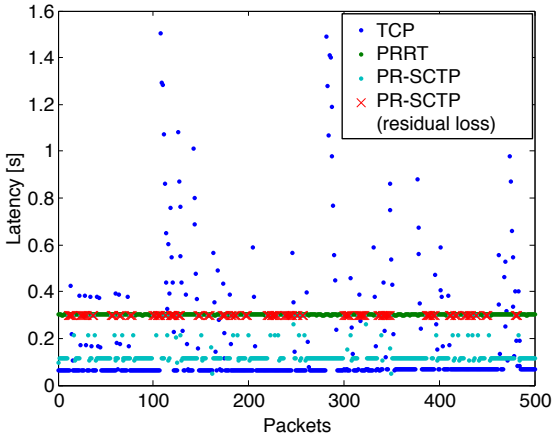


Figure 6.1.: Latency and residual packet loss of ARQ-only error control vs. predictable reliability under a delay constraint ($D_T = 300\,ms$, $RTT = 50\,ms$, $P_e = 0.1$).

Impact of the Round Trip Delay

Table 6.1 shows optimal parameter sets for a 20 $Mbps$ source rate under a delay constraint of $D_T = 300\,ms$ and a target packet loss rate of $P_T = 10^{-5}$. The results are obtained via Algorithm 5.1. The RTT essentially determines the number of reactive repair cycles. Pure packet repetition ($k = 1$) is optimal for lower RTTs and uncorrelated packet loss. At larger RTTs, the collection of longer coding blocks is generally preferred over simple packet repetition, whereas the redundancy information increases along with the RTT. In case of low RTT, the repair packets are just sent reactively upon reception of a NACK ($N_P[0] = 0$) and more repair packets can be shifted to later repair cycles, which occur with exponentially decreasing probability (Section 4.3.1). Consequently, the amount of redundancy is low and close to the theoretical optimum as obtained by Equation 3.12.

Under larger RTT the time budget limits the availability of later repair cycles, i. e. N_C is reduced according to Equation 5.10. PRRT compensates for the missing reactive cycles by sending repair packets proactively ($N_P[0] > 0$). Proactive repair in turn increases the redundancy information (Equation 4.26). In order to amortize the redundancy, PRRT increases the coding block length. In the given example, it achieves optimal performance in the RTT range of 100 ms to 125 ms due to this adaptation. At RTTs larger than 125 ms, the search algorithm reduces the block length again in order to provide sufficient time for one reactive repair cycle under the given delay constraint.

The results show that the predictably reliable transmission under a strict delay constraint depends strongly on the RTT. A purely retransmission-based error control scheme such as the one implemented in TCP or PR-SCTP is insufficient to optimally cover the entire range of the RTT in wide area networks. In particular, those protocols are unable to control the residual packet loss rate if the delivery delay is limited. This functionality is specifically provided by PRRT's reliability control, which adjusts the residual packet loss rate via determining the required amount of redundancy and the optimum repair packet schedule.

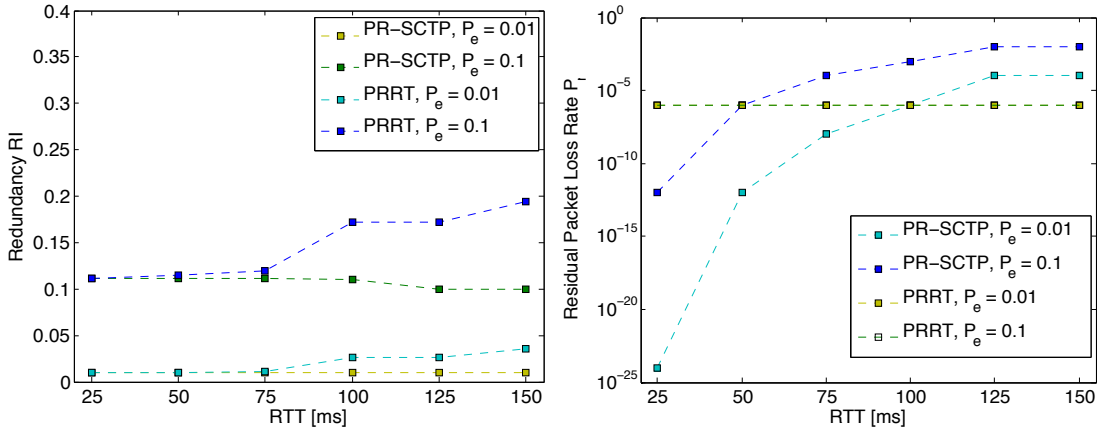


Figure 6.2.: Required redundancy information (left) and residual packet loss rate (right) of PR-SCTP compared to PRRT ($D_T = 300\,ms$, $R_S = 10\,Mbps$)

Impact of Erasure Probability and Temporal Correlation

The packet loss rate primarily determines the number of required repair packets (Table 6.1). As evident from Section 5.1.2, the number of repair packets grows roughly linearly with the coding block length, where the packet loss rate determines the slope. For pure packet repetition under uncorrelated packet loss, the number of repair packets satisfies Equation 5.15. Larger packet loss rates require an increased number of proactive repair packets in order to compensate for the initially higher probability of packet losses in the coding block (Table 6.1). The optimization algorithm finds the parameter set that optimally balances the redundancy caused by the number of proactive repair packets versus the redundancy generated due to the higher probability of sending larger, reactive repair schedules.

Table 6.1 compares optimal coding parameters under i. i. d. and temporally correlated packet loss modeled by a Gilbert-Elliott channel with correlation coefficient $\rho = 0.3$. Longer coding blocks are preferred under temporal correlation, which is beneficial since such configurations amortize the usage of repair packets among a larger number of source packets. As a result of longer block coding the peak-to-average bandwidth requirement of the coded protocol stream is minimized under a bursty packet loss process.

6.1.2. Protocol Goodput

As PRRT follows the Internet path's channel capacity dynamically by adjusting the required amount of redundancy, the ratio between source data and coded data is variable. Therefore, the actual protocol goodput changes over time if the protocol operates under a bandwidth constraint. Besides the protocol-related variables such as header and payload length as well as the feedback frequency, the redundancy information and the desired reliability have significant impact on the protocol goodput (Equation 4.31). The goodput increases as well with the source data rate because of the protocol's higher coding efficiency under higher packet frequency.

Table 6.1.: PRRT configurations for the delivery of a $20\,Mbit/s$ real-time stream with payload size $1316\,byte$ ($D_T = 300\,ms$).

RTT $[ms]$	P_e	ρ	k	N_P	P_r $\times 10^{-6}$	RI
25	0.01	0.0	1	[0, 1, 1, 1]	1.0	0.0101
		0.3	83	[0, 1, 1, 2, 7]	5.6	0.0121
	0.10	0.0	1	[0, 1, 1, 1, 3]	1.0	0.1113
		0.3	83	[3, 4, 4, 6, 17]	6.4	0.1401
50	0.01	0.0	1	[0, 1, 1, 1]	1.0	0.0101
		0.3	23	[0, 1, 2, 6]	4.1	0.0132
	0.10	0.0	1	[0, 1, 1, 4]	1.0	0.1140
		0.3	150	[11, 12, 24]	9.0	0.1584
75	0.01	0.0	1	[0, 1, 2]	1.0	0.0102
		0.3	55	[0, 2, 8]	5.8	0.0202
	0.10	0.0	1	[0, 1, 5]	1.0	0.1500
		0.3	55	[5, 6, 17]	5.7	0.1727
100	0.01	0.0	1	[0, 2]	6.3	0.0200
		0.3	150	[3, 10]	9.8	0.0281
	0.10	0.0	150	[18, 18]	6.6	0.1585
		0.3	150	[22, 25]	7.7	0.1734
125	0.01	0.0	110	[2, 5]	1.8	0.0229
		0.3	110	[2, 10]	4.9	0.0316
	0.10	0.0	110	[14, 15]	9.1	0.1627
		0.3	110	[17, 22]	8.1	0.1843
150	0.01	0.0	41	[1, 4]	9.8	0.0292
		0.3	41	[1, 8]	8.9	0.0441
	0.10	0.0	41	[6, 10]	8.3	0.1916
		0.3	41	[8, 16]	8.1	0.2384

Redundancy Information

PRRT's goodput evolves reciprocally to the required redundancy information (Equation 4.31), which is examined in the following. As observed from the example parameter set in Table 6.1, it modifies the redundancy in the coding process in response to the network state such that under constrained delay the reliability above transport layer remains constant with high probability. Figure 6.3 shows PRRT's redundancy profile under a source rate of $20\,Mbps$ and i. i. d. packet loss. In compliance with the definition of the erasure channel's capacity (Equation 3.3), the redundancy is basically determined by P_e (Equation 3.12). However, as the RTT determines the granularity and consequently the efficiency of the adaptive HEC scheme, it also increases along with a larger RTT.

Similarly, the redundancy increases along with an increasing receiver group size as packet losses are assumed to be independent among the receivers of the group. Figure 6.3 (center) shows the redundancy profile for a group of 10 receivers. For larger receiver groups the error control becomes more proactive, independently from the RTT. As a result, the redundancy is constantly higher than for a single receiver. The flat shape of the graph shows that the amount of redundancy is almost entirely determined by P_e.

Figure 6.3 (bottom) presents the redundancy profile under temporal correlation. The redundancy increases significantly under low RTT since the protocol must increase the amount of proactive error control even under those circumstances where short coding blocks and reactive repair are optimal under independent packet loss. For i. i. d. packet loss, the redundancy is a concave function of the RTT (Figure 6.3, top). However, larger correlation coefficients turn it into a convex function of the RTT.

The source data rate affects the efficiency of both the proactive and the reactive redundancy. Figure 6.4 shows the amount of redundancy information for a fixed RTT of $100\,ms$ and a source rate ranging from 10 to $30\,Mbps$. It is visible that the required redundancy information is inversely proportional to the source data rate. This is because larger source data rates allow for more efficient protocol configurations with longer coding blocks due to their shorter packet interval.

Goodput vs. Reliability

PRRT's parameters are optimized under the objective of minimizing the required redundancy information (Section 5.1.1). However, if a multimedia stream with rigid source rate must be delivered under a limited network bandwidth, the redundancy might be limited such that the objective changes into the maximization of the reliability under the given bandwidth constraint. In the following, the goodput is therefore formulated in dependence on the desired residual packet loss rate.

Figure 6.5 presents PRRT's goodput under the following assumptions: The payload size is $L_D = 1316\,byte$, the source header size $L_{DHdr} = 24\,byte$ and the repair header size $L_{PHdr} = 16\,byte$. Further, a rate limit of $R_C = 20\,Mbps$ is assumed for the experiment. The goodput is evaluated under variable RTT, packet erasure rate P_e and correlation coefficient ρ of the simplified Gilbert-Elliott model. Under a given rate limit, goodput can be traded for reliability in a small range that depends on P_e and ρ.

For a comparison, the theoretical goodput under a residual erasure rate P_r and a measured erasure rate P_e is obtained as

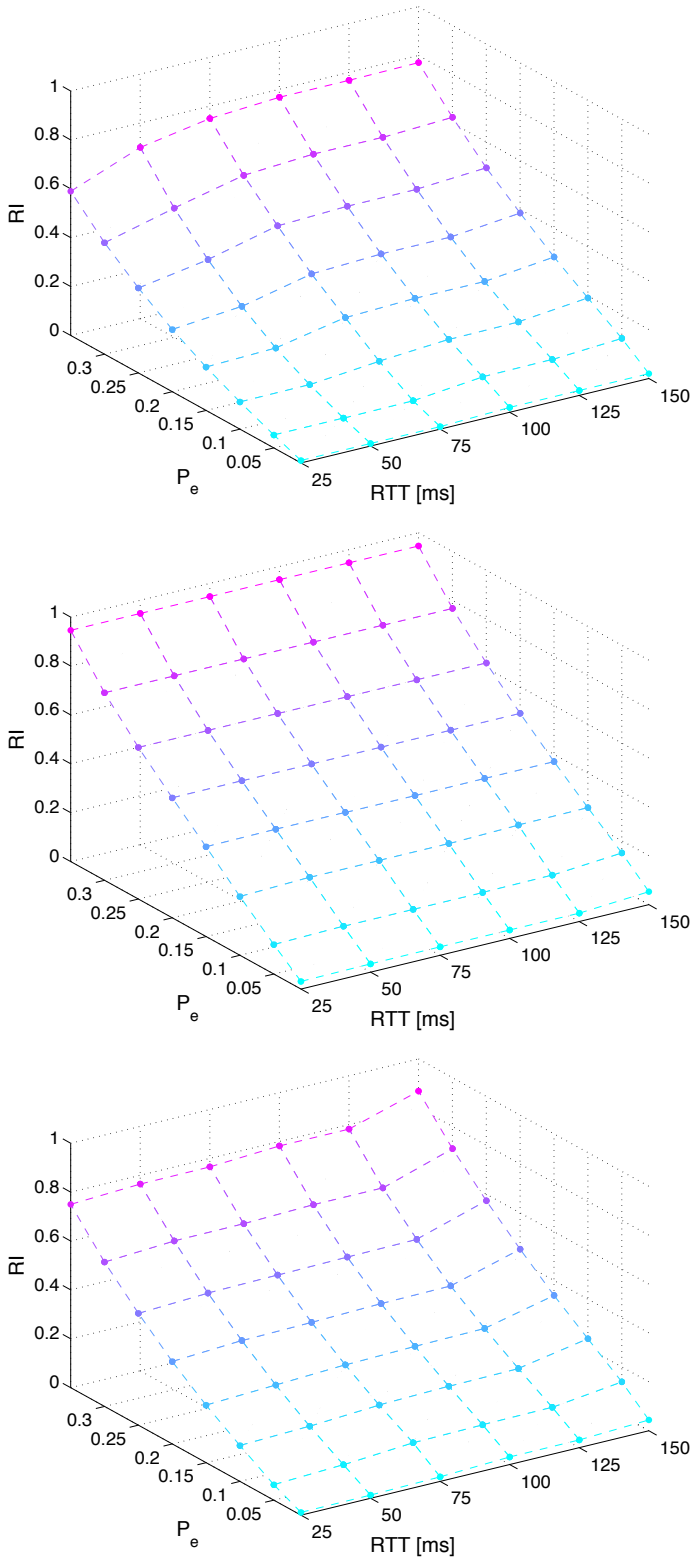


Figure 6.3.: Redundancy profile depending on erasure rate and RTT ($D_T = 300\,ms$); $\rho = 0.0$, 1 receiver (top), 10 receivers (center), $\rho = 0.3$, 1 receiver (bottom).

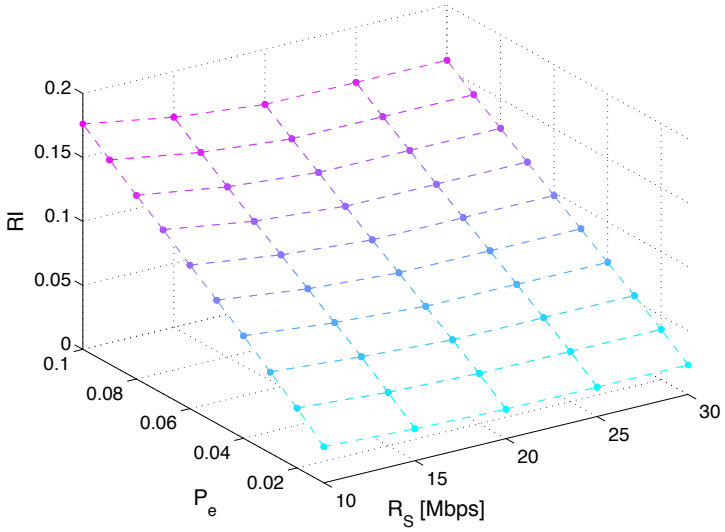


Figure 6.4.: Redundancy profile depending on erasure rate and source rate ($D_T = 300\,ms$, $RTT = 100\,ms$)

$$R_{S,opt} = \frac{R_C}{1 + RI_{opt}} = \frac{R_C}{1 + \frac{P_e - P_r}{1 - P_e}} \tag{6.1}$$

based on Equation 3.12. PRRT's goodput is compared with the theoretical optimum $R_{S,opt}$ in Figure 6.5 (dotted line). In the practical protocol implementation, where code length and time budget are limited, two additional parameters affect the goodput. First, Figure 6.5 shows that an increased correlation ρ in the packet loss process requires a substantial increase in the added redundancy information. This is especially pronounced at higher P_e, which is explained by the need to decrease the code rate in order to compensate for packet losses with higher burstiness. Second, a larger amount of redundancy is also added if an increased RTT limits the granularity of the protocol configuration.

6.2. Internet Media Streaming

The majority of today's video streams is delivered via HTTP while suffering from severe quality reduction under TCP's inefficiencies in such application scenarios. High-quality, real-time video streaming is therefore still restricted to the managed Internet, which provides guaranteed QoS to IP-based multimedia applications. Yet managed flows require support from the underlying infrastructure and rely on individual service level agreements with the corresponding service provider. Hence, QoS guarantees are missing for flows that are delivered beyond the service provider's infrastructure as well as over lossy home network segments.

Outside of managed infrastructure, reliability and multiple access are ensured by self-managed, end-to-end error and congestion control on transport layer. However, available transport layer protocols optimize their objectives in error and congestion control without respect to the application's individual QoS constraints such that multimedia services suffer from significant degradation. Nevertheless, congestion control is a crucial compo-

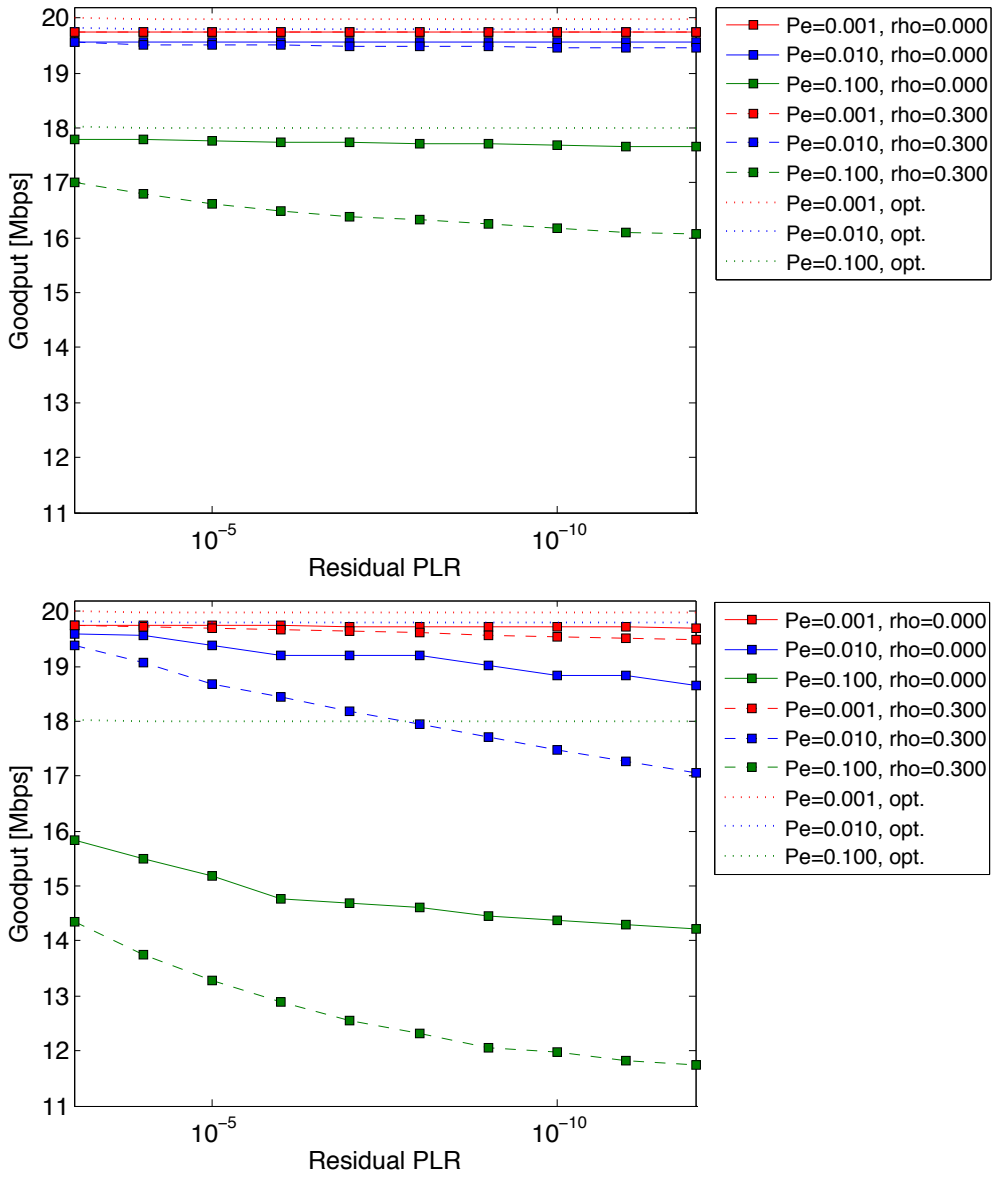


Figure 6.5.: Protocol goodput under $D_T = 400\,ms$, $RTT = 50\,ms$ (top) and $D_T = 400\,ms$, $RTT = 150\,ms$ (bottom)

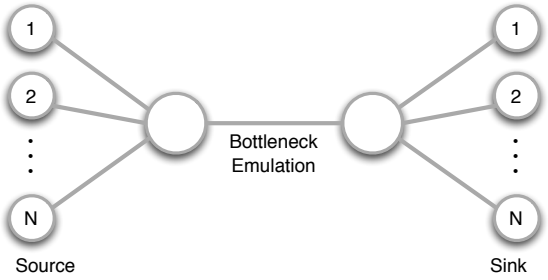


Figure 6.6.: Dumbbell topology.

nent of any reliable transport protocol operating on the unmanaged Internet. Therefore, dynamic stream switching has been proposed to provide smooth video distribution with quantized quality adaptation under the Internet's multiplexing policy.

The following section demonstrates, how streaming video is delivered via PRRT while fulfilling nearly broadcast-compatible QoS requirements while respecting the objectives of a stable and fair usage of the Internet bandwidth. PRRT is therefore configured to perform under congestion losses, is enhanced by a delay-based congestion control equation, and finally integrated into a dynamic streaming application.

6.2.1. TCP-induced Packet Loss

The Internet's typical source of packet loss is congestion. Unfortunately, congestion is deliberately induced by loss-based congestion control such that queue overflows happen frequently. Continuous packet streams suffer especially under the congestion control policy of the Internet's dominant transport layer protocol, TCP, which aggressively fills network queues until it observes their overload from packet loss. The queueing loss particularly affects non-elastic multimedia streams that compete with TCP's loss-based congestion control at a network bottleneck. Therefore, this observation is commonly known under the term *TCP-induced packet loss*.

Experimental Setup

In order to evaluate PRRT's performance under congestion loss, it is sent through a network bottleneck along with a different number of TCP sessions. The experimental setup comprises a dumbbell topology with a 50 $Mbps$ bottleneck bandwidth emulated via a Dummynet[1] bridge (Figure 6.6). The base RTT of the Dummynet bridge is set to 50 ms, 100 ms and 150 ms during different tests. The emulator performs drop-tail queueing, whereas the buffer size is set to the bandwidth-delay product of the emulated link. In different experiments, PRRT streams of 5 $Mbps$, 10 $Mbps$ and 20 $Mbps$ source rate are sent through the bottleneck along with several TCP-Cubic [68] sessions established via Iperf[2]. During several experiments, PRRT competes with up to 9 TCP streams while their number and PRRT's source rate are adjusted to achieve an equal share of the bandwidth for all flows. All experiments are terminated after transmitting 10^7 PRRT packets with a payload size of 1316 $byte$. The response delay of the hosts is set to

[1] http://info.iet.unipi.it/~luigi/dummynet/
[2] http://sourceforge.net/projects/iperf/

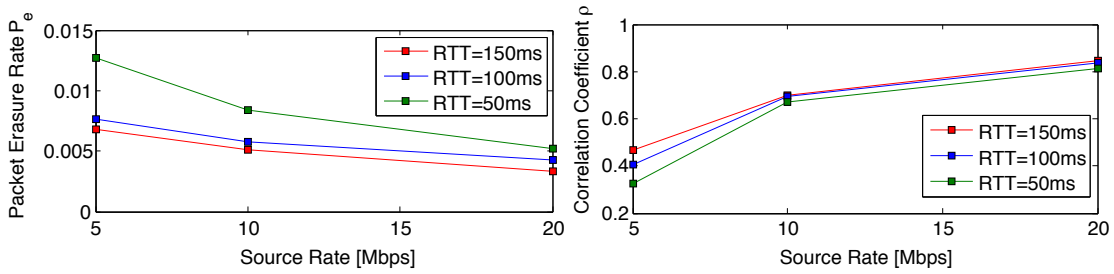


Figure 6.7.: Measured packet erasure rate and correlation coefficient obtained by fitting a Gilbert-Elliott model to the observed average burst and gap length under congestion loss.

$D_{RS} = 20\,ms$ and the reliability constraint to a residual packet loss rate of $D_T = 10^{-5}$. PRRT adapts with an update interval of $T_{UPD} = 200\,ms$ under a feedback interval of $T_{NFB} = 100\,ms$. For comparison the experiments are repeated while replacing PRRT by PR-SCTP as a representative for a partially reliable transport protocol.

Network State under Queue Saturation

PRRT measures the network state in terms of the packet loss rate P_e, the burstiness coefficient of the loss process ρ as well as the network path's round trip time RTT. Those parameters characterize the impact of the TCP-induced queue saturation on the continuous media stream and they essentially determine PRRT's adaptive parametrization. However, during the experiments the network state is dynamic such that some characteristic long-term trends are discussed in the following. The observations are represented via their average values obtained throughout the entire experiment.

The erasure probability and the burstiness of the packet loss process observed by the PRRT stream are influenced by two parameters (Figure 6.7): PRRT's source rate as well as the network's RTT. The erasure probability implicitly reflects the frequency of the congestion events, which grows for lower RTTs, where TCP's congestion window oscillates faster. The burstiness of the packet loss refers to the impact of a congestion event on the real-time flow, which mainly increases together with the source rate.

The temporal correlation in the packet loss is expressed by instantiating the protocol's block-erasure model as a simplified Gilbert-Elliott model (Section 3.2.2). The model has previously been proposed to express the burstiness of measured packet loss on Internet paths [21, 166]. It relies on the assumption that periods of packet loss and periods of successful reception have geometrically distributed length. The model is fit via maximum likelihood estimation (Section 3.2.2) to the observed loss indicator sequence (Section 2.4.2) in order to obtain the model parameters consisting in packet erasure rate P_e and correlation coefficient ρ.

The GE model's average correlation coefficients for different test scenarios are shown in Figure 6.7 (right). It is visible that the RTT has minor impact on the correlation coefficient. At high sending rates the model compensates for large average erasure lengths with a high correlation coefficient ρ. The measurements perfectly support the simplified GE model at sending rates around $5\,Mbps$ (Figure 6.8). However, according to the results it does not hold for real-time streams at rates beyond $10\,Mbps$ under the available measurement setup. Two effects are deemed to be responsible for these findings: First,

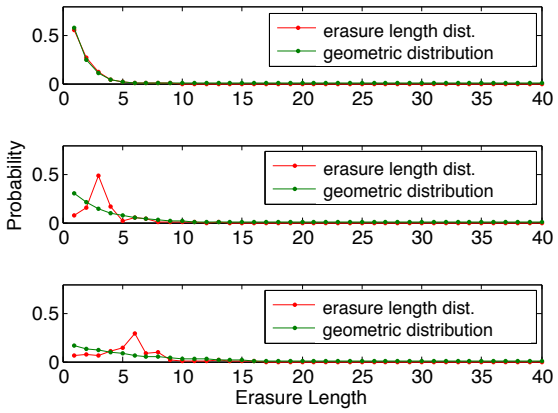


Figure 6.8.: Probability mass function of the erasure length for the source rates $R_S = 5\,Mbps$ (top), $R_S = 10\,Mbps$ (center) and $R_S = 20\,Mbps$ (bottom) (truncated at 40 packets, $RTT = 100\,ms$).

higher sending rates lead to a larger number of packets being blocked consecutively at the saturated queue. Figure 6.8 supports this assumption by showing a mean erasure length of greater than 3 at $10\,Mbps$ and greater than 6 at $20\,Mbps$ sending rate. Second, TCP sessions that are sharing the same bottleneck tend to acquire *global synchronization* [157], which results in longer periods of network saturation and significant underutilization afterwards, due to their collective window reduction. TCP synchronization is known to increase for a lower number of parallel TCP sessions. For higher media source rates this leads to a larger number of packets being blocked consecutively at the saturated queue.

The GE model compensates for the large average erasure length with an overestimated correlation coefficient ρ. A model of higher order or a queueing model would be more appropriate to express the temporal correlation of queueing losses. However, the overestimation of the correlation coefficient results in the selection of conservative protocol parameters by PRRT's reliability control unit. Therefore, the significant increase in complexity introduced by a more sophisticated network model is not justified for the real-time application.

QoS Allocation via PRRT

Table 6.2 concludes selected protocol configurations obtained under the characteristic average values from Figure 6.7. PRRT's reliability control finds the optimal block length k and a corresponding repair packet schedule N_P subject to a desired residual erasure rate P_T and a delay constraint D_T (Section 5.1.1). This results in a dynamic code rate obtained by the parameter adaptation itself as well as the incremental sending of repair packets. Pure FEC represents the borderline case of sending all repair packets proactively. Purely reactive repair with a block length of $k = 1$ corresponds to the functionality of an ARQ-based, partially reliable protocol (Figure 2.1).

Because of the fact that the continuous media stream cannot instantly back off the sending rate at the TCP-induced congestion event, packet erasures tend to appear in longer sequences due to the temporally correlated queueing losses. Reactive repair packets must not concentrate on a single repair packet cycle immediately after sensing the

Table 6.2.: Selected PRRT configurations under congestion loss with different RTTs and source rates R_S ($P_T = 10^{-5}$).

RTT $[ms]$	R_S $[Mbps]$	P_e	ρ	D_T $[ms]$	k	N_P	RI
50	5	0.013	0.82	300	16	[0, 2, 7]	0.0247
	10	0.009	0.67		32	[0, 5, 18]	0.0219
	20	0.006	0.33		64	[0, 10, 31]	0.0143
100	5	0.008	0.41	500	16	[0, 2, 8]	0.0158
	10	0.006	0.70		32	[0, 6, 17]	0.0149
	20	0.005	0.84		64	[0, 11, 34]	0.0192
150	5	0.007	0.47	700	16	[0, 3, 8]	0.0145
	10	0.005	0.70		32	[0, 6, 17]	0.0131
	20	0.004	0.85		64	[0, 11, 34]	0.0090

packet loss as they would contribute to the queue saturation. Therefore, a small multiple of the RTT should be available for PRRT's time budget D_T in order for the search algorithm to spread the repair packets over several cycles (Table 6.2). Alternatively, if the RTT is large compared to D_T, the protocol can operate as an adaptive FEC at the price of reasonably larger coding overhead. Whereas an FEC configuration requires 60% to 70% coding overhead to satisfy the reliability requirement of $P_T = 10^{-5}$, hybrid configurations add less than 3% overhead within all considered scenarios (Table 6.3).

As a metric for PRRT's QoS allocation, the results are compared with the requirements of specific ITU-T Y.1541 QoS classes. Class 6 and class 7 formulate the tightest reliability constraint with a residual packet loss rate of 10^{-5} under an end-to-end delay of $100\,ms$ and $400\,ms$, respectively. Those requirements approach the QoS constraints of IP-based live media broadcast. As evident from Table 6.3, class 6 can only be satisfied by an adaptive FEC configuration under low RTT, whereas the requirements of class 7 are met under larger RTT. For RTT's around $50\,ms$, even the more efficient hybrid (proactive and reactive) configuration meets the constraints of class 7. For RTTs of $100\,ms$ and more, however, the delay constraint of $400\,ms$ is too tight for the hybrid error control such that these configurations fulfill just class 4 with a delay constraint of $1\,s$.

Finally, the results of PRRT are compared with the correction performance of PR-SCTP under the same time constraints (Table 6.3). Unfortunately, it is hardly possible to implement a fair comparison between both protocols since PR-SCTP's performance significantly suffers under the combination of loss-based congestion control via AIMD and ARQ-only error control. As a result of the incompatibility of both schemes with real-time media transport, a large amount of packets is rejected under the specified delay constraints. In the evaluated scenario, PR-SCTP achieves therefore residual packet loss rates between 1 % and 16 % during the competition with TCP at the network bottleneck.

6.2.2. Congestion Control

PRRT is prepared for equation-based congestion control (Section 5.3.1). During the following experiments, PRRT is equipped with a delay-based control equation derived from the FAST TCP approach [160]. Delay-based congestion control has specifically been proposed for network paths with a large bandwidth-delay product, where the packet

Table 6.3.: PRRT's residual packet loss P_r and coding overhead RI under different round trip time RTT, source rate R_S and delay constraint D_T ($P_T = 10^{-5}$). Assignment of the corresponding ITU-T Y.1541 QoS classes and comparison with PR-SCTP.

RTT $[ms]$	R_S $[Mbps]$	D_T $[ms]$	P_T	P_r $[\times 10^{-2}]$	RI	QoS Class
PRRT						
50	5	100	10^{-5}	0.0009	0.562	6
	10			0.0001	0.716	
	20			0.0000	0.640	
100	5	150		0.0012	0.620	7
	10			0.0000	0.719	
	20			0.0000	0.705	
150	5	200		0.0012	0.697	7
	10			0.0000	0.721	
	20			0.0007	0.713	
50	5	300	10^{-5}	0.0004	0.025	7
	10			0.0007	0.022	
	20			0.0010	0.015	
100	5	500		0.0000	0.016	4
	10			0.0011	0.015	
	20			0.0010	0.012	
150	5	700		0.0011	0.015	4
	10			0.0000	0.013	
	20			0.0012	0.010	
PR-SCTP						
50	5	300	n. a.	16.1184	0.001	n. a.
	10			2.8925	0.001	
	20			1.3458	0.001	
100	5	500		13.7375	0.002	
	10			6.2276	0.001	
	20			1.2907	0.001	
150	5	700		6.1445	0.003	
	10			2.5706	0.002	
	20			1.6592	0.002	

frequency is particularly high. Since high-quality multimedia applications offer a continuously high packet rate to the network, this condition is fulfilled. Under the large packet frequency the delay-based control equation obtains RTT samples in a sufficiently high density such that the queueing delay provides a reliable congestion signal.

Control Equation

The FAST TCP equation [160] is extended by an additional weighting factor β as follows [67]:

$$\begin{aligned} R(t) &= (1-\gamma)\cdot R(t-1) \\ &+ \gamma\cdot\left((1-\beta)\cdot\frac{baseRTT}{avgRTT}\cdot R(t-1)+\alpha\cdot\beta\right), \end{aligned}$$

where $\alpha > 0$ and $\beta, \gamma \in (0,1)$. $R(t)$ and $R(t-1)$ are the most recent and the previous estimate of the available bandwidth, respectively. $baseRTT$ is the minimum RTT observed so far and $avgRTT$ is the current average RTT after applying EWMA. The control equation itself performs EWMA with parameter γ over current and previous rate estimates. α reflects the number of packets buffered in the network, which should not exceed the network's queue size. Both parameters originate from the FAST TCP equation. β provides control over the protocol's aggressiveness in bandwidth acquisition as a larger value for β attenuates the congestion signal. As a result, the protocol acquires available bandwidth with controllable fairness towards background traffic depending on the choice of β. A larger value of β gives priority to the PRRT stream. The experiments presented in this thesis are obtained with $\alpha = 180$, $\beta = 0.1$ and $\gamma = 0.5$.

Shared Long-distance Internet Path

The functionality of PRRT extended by the delay-based control equation is demonstrated at the example of a long-distance Internet path. Specifically, Internet paths with large bandwidth-delay product are a demanding environment for most congestion control algorithms because the reaction to congestion events is slow as a result of the large RTT. In addition, the congestion window has to grow large to utilize the available bandwidth.

In the following, PRRT is sent through an Internet connection between Saarbruecken, Germany and Los Angeles, USA. The path includes 23 hops and has a measured $baseRTT$ of $160\,ms$. Throughout the whole experiment, the PRRT stream is sent along with one parallel TCP-Cubic session. The TCP stream is evaluated as an indicator for PRRT's TCP-fairness. Three additional TCP streams enter the network path with a delay of $60\,s$ each. The latest TCP stream is active for $60\,s$. Afterwards, the three TCP streams are terminated in reverse order, where one stream stops every $60\,s$. The whole experiment runs for $420\,s$. PRRT is configured with a payload size of $1316\,byte$, a delay constraint of $D_T = 300\,ms$ and a reliability requirement of $P_T = 10^{-5}$.

The measured throughput of PRRT and the competing TCP sessions is depicted in Figure 6.9 (left). The PRRT stream achieves significantly smoother throughput due to the delay-based control equation that calculates an average over the network's queueing delay. Similarly, the equation reacts smoothly to the increased load on the network path induced by the additional TCP streams. As indicated by the average throughput of the first TCP stream that shares the path during the whole experiment, PRRT reduces

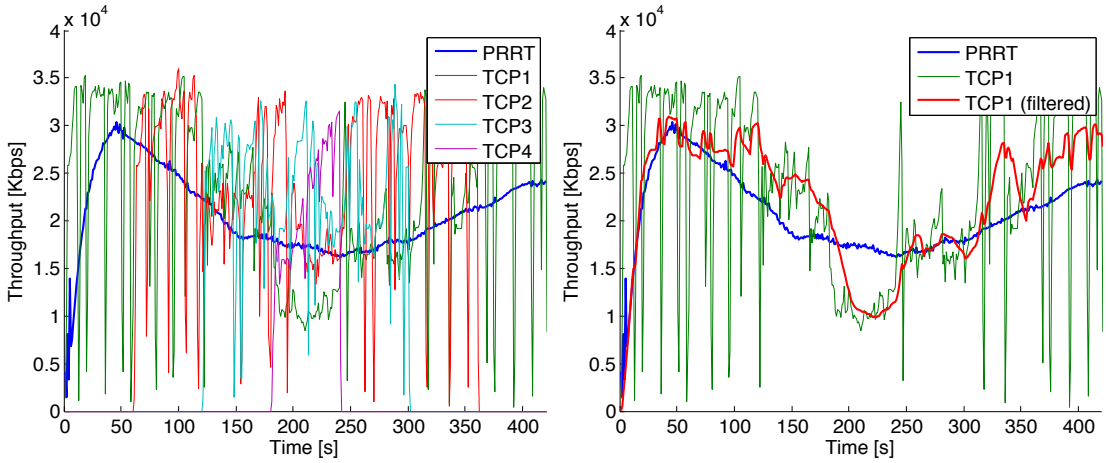


Figure 6.9.: Comparison of PRRT with delay-based congestion control and TCP-Cubic on a network path between Saarbruecken (Germany) and Los Angeles (USA) with $RTT = 160\,ms$.

the throughput very similarly and in the same order of magnitude as the TCP stream does (Figure 6.9, right, filtered TCP). Both streams obtain a fair share of the available bandwidth as indicated by Jain's index of $\mathcal{J} = 0.998$ while $\mathcal{J} = 1$ refers to optimum fairness [82].

During the experiment the PRRT stream experiences an average packet loss rate of 0.0053, whereas the peak packet loss rate is observed as 0.0285 during a congestion event. The application experiences a residual packet loss rate of $P_r = 6.64 \cdot 10^{-6}$ throughout the experiment, which fulfills the reliability constraint of $P_T = 10^{-5}$. The average redundancy information added to the source packet stream measures $RI = 0.0079$.

Loss Differentiation and Opportunistic TCP-friendliness

Loss-based congestion control suffers under the presence of packet loss caused by physical packet corruption, which is particularly frequent in wireless networks. The result is a significant underestimation of the available bandwidth due to unnecessary window reductions upon erroneously treating such packet losses as congestion signals. Despite the fact that wireless and mobile networks implement error control on the MAC layer, transport layer protocols are still exposed to a small fraction of residual packet loss as those schemes do not correct exhaustively.

Delay-based congestion control is insensitive to packet loss such that the protocol can maintain a stable throughput even on wireless network infrastructures with physical packet loss [70]. Moreover, as the control equation continuously evaluates the network's queueing delay, it detects an underutilization of the available bandwidth such that it increases its rate allocation beyond a TCP-friendly amount in case TCP is unable to obtain an efficient share. This behavior is defined as *opportunistic TCP-friendliness* [156].

The insensitivity to packet loss and the opportunistic TCP-friendliness of PRRT are evaluated on a physical network infrastructure with an emulated bottleneck bandwidth of $32\,Mbps$ and an RTT of $50\,ms$. The bottleneck and the delay are emulated via a

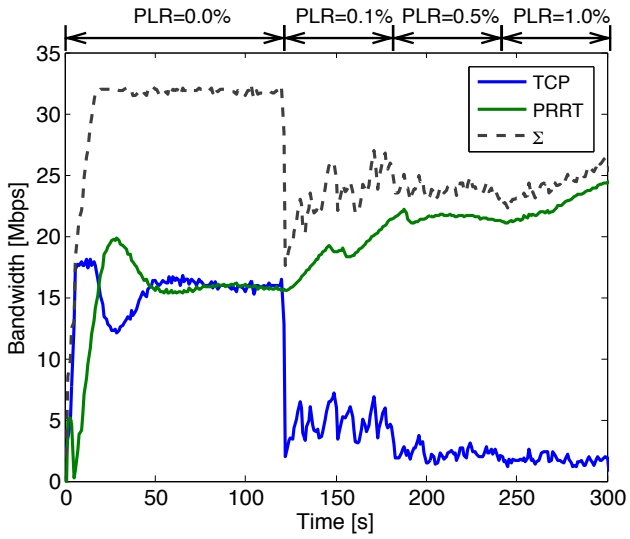


Figure 6.10.: Opportunistic TCP-friendliness under delay-based congestion control. Emulated packet loss rate, bottleneck bandwidth 32 $Mbps$, $RTT = 50\,ms$.

Dummynet bridge with a queue size of 200 KB. The opportunistic bandwidth allocation of the PRRT protocol is measured under the delay-based congestion control in presence of different packet loss rates while competing for bandwidth with a TCP-Cubic stream. The experiment evaluated in Figure 6.10 runs without physical packet loss during the first 120 s. Afterwards, an emulated packet loss rate is applied via the Netem emulator[3]. The emulated packet loss is increased every 60 s to the following rates: 0.1 %, 0.5 % and 1 %. Both PRRT and TCP-Cubic obtain network bandwidth greedily in this experiment. The TCP session is established via Iperf.

As evident from Figure 6.10, the network bottleneck remains saturated as long as the network path experiences exclusively queueing losses induced by TCP's loss-based congestion control. Both protocol sessions obtain a fair share of the available bandwidth. Already under a small additional packet loss rate of 0.1 % the TCP throughput reduces to roughly 25 % of the fair bandwidth share. Due to the delay-based congestion control, PRRT's throughput is not affected by the packet loss. Moreover, the control equation immediately detects the underutilization of the bandwidth and carefully increases the throughput. Under a physical packet loss rate of 1 %, the delay-based congestion control leads to more than 8 times the throughput of TCP-Cubic without negatively affecting the TCP session since the network is not saturated.

6.2.3. Dynamic Media Streaming

In the following PRRT's QoS allocation and the delay-based congestion control are combined under a dynamic streaming application. This example proposes an alternative architecture of dynamic video streaming based on predictable reliability [67]. In fact, Internet TV solutions based on HTTP are limited in their quality due to several reasons. First, TCP's error control is purely based on ARQ, which does not allow to deliver inelas-

[3] http://www.linuxfoundation.org/collaborate/workgroups/networking/netem

tic traffic with limited latency [122]. Second, the congestion control policy of loss-based AIMD introduces severe goodput variations such that smooth media streaming is only possible if roughly twice the multimedia bandwidth is available [158]. Third, the protocol performs congestion control in an application-agnostic fashion [141]. TCP transmits chunks of the media stream in a pull mode, clocked by the strobe of receiver acknowledgments such that the application has very limited influence on the sending rate. As a result, HTTP-based media transport relies on highly conservative goodput estimates and extensive buffering in the receiving host in order to overcome TCP's shortcomings.

Related Work in Dynamic Media Streaming

Related work in the area of streaming media delivery addresses mainly the following issues: Inelastic media streams introduce congestion if not being under proper packet scheduling and rate control. On the other hand, they are subject to congestion losses introduced by the widely TCP-based background traffic. On wireless networks, physical interference and protocol collisions lead to additional packet erasures. Therefore, related work in this field commonly builds upon *dynamic streaming, packet scheduling* and *source rate adjustment* of media streams as well as *partial reliability, loss tolerance* and *loss differentiation* on transport layer.

Recent developments in dynamic HTTP streaming try to address the issues of media streaming under congestion control by offering the video stream in multiple resolutions at the server [74]. This technique has gained much interest in commercial video streaming solutions such as Apple HTTP Live Streaming[4], Microsoft Smooth Streaming[5] and Adobe HTTP Dynamic Streaming[6] [5]. In every adaptation cycle the version of the video is chosen that fits the current congestion window. As a part of the 3GPP[7] project, Dynamic Adaptive Streaming over HTTP (DASH) [144] is in its standardization process. Extensive modeling and evaluation of suchlike schemes has, however, unveiled that they achieve roughly half bandwidth utilization under buffering of several seconds [5, 158].

A stream switching scheme similar to DASH has been developed by De Cicco et al. [47]. The authors monitor the sending buffer at the streaming server and formulate the selection of the quality level as a control law of the buffer level. Congestion-distortion optimized scheduling [138] gives priority to key frames of video streams and optimizes the sending schedule of predicted frames in order to avoid queues building up at a network bottleneck. In case of congestion, predicted frames might be dropped as they have minor impact on the picture quality. Combined control of source and channel coding has been proposed in the Multimedia Streaming TCP-Friendly Protocol (MSTFP) [172]. The scheme allocates network bandwidth proactively with a network-adaptive rate control scheme using equation-based congestion control. Source and channel coding are instantly optimized under the observed network state.

TCP's abrupt window reduction has been identified to negatively interfere with continuous streaming services. Therefore, the Video Transport Protocol (VTP) [156] has been designed so as to reduce the sending rate in response to a congestion signal only to the incoming data rate measured at the receiver instead of halving the congestion window. Besides loss differentiation, VTP implements opportunistic TCP-friendliness,

[4] http://tools.ietf.org/html/draft-pantos-http-live-streaming-08
[5] http://www.iis.net/download/smoothstreaming
[6] http://www.adobe.com/products/hds-dynamic-streaming.html
[7] http://www.3gpp.org/

which allows the protocol to acquire unutilized bandwidth as long as the throughput of parallel TCP sessions is not affected.

On transport layer the proposed solutions concentrate on dedicated extensions to TCP's error and congestion control in order to increase TCP's throughput under physical packet loss by adding loss tolerance (LT-TCP [150]) or loss differentiation (TCP-Westwood+ [101, 46]). For rate-controlled media streaming without reliability the Datagram Congestion Control Protocol (DCCP) has been proposed based on TCP-friendly Rate Control (TFRC) [163]. A Partial reliability extension (PR-DCCP [90]) has been specified for this protocol by allowing for a limited number of packet repetitions. Partial reliability is also implemented into PR-SCTP [128], leading to similar correction performance. Since all of the above transport protocols implement window-based, TCP-like congestion control, they considerably reduce the throughput in presence of large delay variations or physical packet loss in the delivery network.

Experimental Setup

The following experiments pronounce the increase in video quality obtained under the PRRT-based transport in dynamic media streaming, in particular, in presence of network congestion and physical packet corruption. The application is implemented according to the architecture proposed in Figure 5.7. First, the response of the PRRT-based dynamic streaming to network congestion is demonstrated under competition with TCP background traffic at a network bottleneck. Afterwards, the stand-alone performance of the scheme is evaluated on an emulated wired broadband connection terminated by a physical wireless home network as well as a physical mobile broadband network. All results are compared with a reference implementation of DASH based on TCP-Cubic.

For the dynamic stream switching, a video file is available in two adaptation sets with a chunk size of $2\,s$ at the streaming server. The first adaptation set includes bit rates of $0.8\,Mbps$, $1.5\,Mbps$ and $2.5\,Mbps$, the second set consists of $2\,Mbps$, $3\,Mbps$, $4\,Mbps$, $5\,Mbps$ and $6\,Mbps$. The DASH reference design, implemented into a plugin[8] for the VLC[9] video player, is applied for the comparison. The DASH plugin is configured with a buffer size of $5\,s$. In the PRRT-based streaming architecture, this value limits the time budget for the error control. Furthermore, the protocol is configured to achieve a residual packet loss rate of $P_T = 10^{-5}$ and to carry a payload size of $L_D = 1316\,byte$. The update interval of the reliability control is set to $T_{UPD} = 200\,ms$ and the interval of the network state feedback is $T_{NFB} = 100\,ms$.

Response to Congestion

The behavior of both dynamic streaming based on PRRT and TCP is evaluated while competing with an increasing number of TCP-Cubic sessions at a network bottleneck. The bottleneck is emulated by a Dummynet bridge in a dumbbell topology (Figure 6.6) with a bandwidth of $16\,Mbps$, a one-way propagation delay of $35\,ms$ and a queue size of $140\,KB$. In both experiments, the dynamic streaming architecture runs $120\,s$ without background traffic. Afterwards, 4 TCP sessions emulated via Iperf enter the network successively with an offset of $60\,s$ each. $360\,s$ after the beginning of the experiment, the competing TCP streams are terminated.

[8] http://www-itec.uni-klu.ac.at/dash/
[9] http://www.videolan.org/

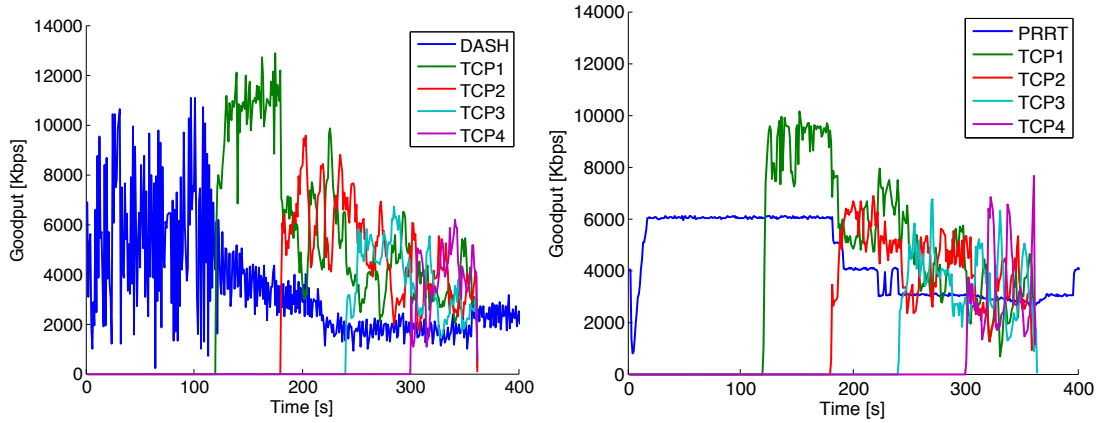


Figure 6.11.: Goodput of the dynamic video streaming via DASH based on TCP-Cubic (left) and based on PRRT (right).

Figure 6.11 (left) shows the goodput of the DASH reference implementation obtained for the test video in the second adaptation set (2 $Mbps$ to 6 $Mbps$). During the absence of background traffic the scheme's goodput oscillates as a result of TCP's throughput variations and the loss-based congestion control. However, the buffer of 5 s at the receiver is still sufficient to render the video smoothly at the highest quality version. As soon as background traffic enters the network, the DASH setup significantly reduces the video rate and cannot allocate a fair share of the bottleneck bandwidth. In parallel to 4 TCP sessions, the DASH scheme drops the sending rate to the minimum video bit rate of 2 $Mbps$.

PRRT's bandwidth allocation is evaluated in Figure 6.11 (right). The protocol maintains a constant goodput at the maximum video bit rate of 6 $Mbps$ after a short convergence phase of the congestion control equation. At the time the first competing TCP stream enters the bottleneck, PRRT's bandwidth allocation stays constant as the sender's packet scheduling is limited by the video rate to less than the fair share of the bottleneck bandwidth. As the number of competing TCP streams increases, PRRT reduces the goodput such that lower quality versions of the video must be chosen. However, as a result of the smooth control equation, the throughput does not oscillate. Therefore, the video streaming can be performed at a higher quality version compared to the TCP-based DASH.

Wireless Home Network

In a further experiment the dynamic video streaming via PRRT is evaluated over a wired broadband connection with a wireless home network segment. The wireless receiver is positioned in a distance of 7 m to the access point (AP) with two walls on the direct signal path. The wired broadband access is emulated via a Dummynet bridge that implements queueing and introduces a base RTT of 50 ms between the streaming server and the AP. The AP of the wireless local network additionally introduces extensive queueing delays that are caused by the dynamic frequency selection (DFS) scheme defined within the wireless standard. This scheme requires the channel to be scanned periodically for radar sequences and interfering signals, which leads to packets being queued up at the AP

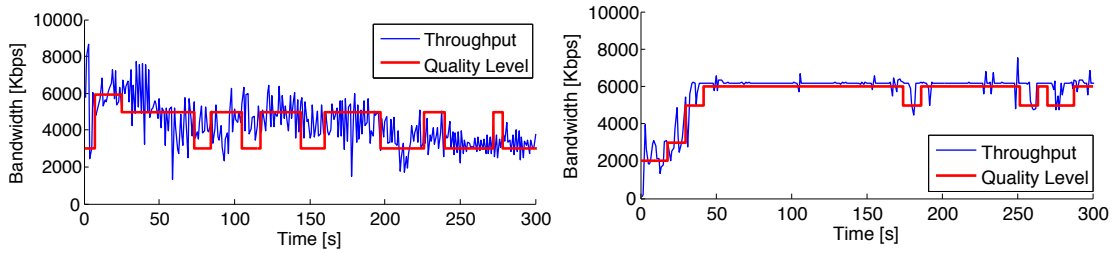


Figure 6.12.: Throughput and quality level of dynamic video streaming via DASH on TCP-Cubic (left) and based on PRRT (right). Underlying network infrastructure is an IEEE 802.11n wireless LAN.

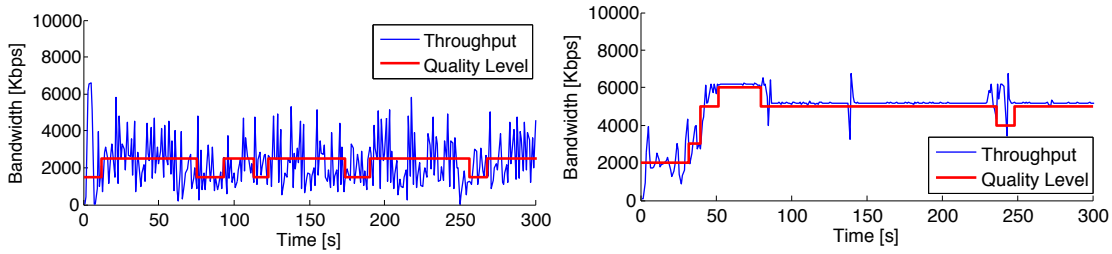


Figure 6.13.: Throughput and quality level of dynamic video streaming via DASH on TCP-Cubic (left) and based on PRRT (right). Underlying network infrastructure is HSDPA with 7.2 *Mbps* downlink.

because the tuner is unavailable during these checks.

Throughout the experiment, TCP generally achieves a sufficiently high throughput on the wireless local network in order to apply the second adaptation set (2 *Mbps* to 6 *Mbps*). TCP-Cubic's aggressive window control leads the DASH implementation to switch the video bit rate abruptly in steps of up to two quality levels (Figure 6.12, left). The PRRT-based streaming mostly achieves sufficient throughput to transmit the video at the highest bit rate of 6 *Mbps* (Figure 6.12, right). Table 6.4 demonstrates the gain of throughput under PRRT compared to DASH via the data volume that is transmitted over the whole session. The presented architecture increases the throughput by more than 30 % compared to the TCP-based setup.

Table 6.4.: Throughput gain of PRRT-based dynamic streaming compared to DASH on TCP-Cubic.

	WLAN		3G	
	DASH	PRRT	DASH	PRRT
Adaptation Set	2	2	1	2
Mbytes sent	162.43	215.55	90.33	184.46
Throughput Gain	–	32.71 %	–	104.22 %

Mobile Broadband Network

Finally, the resulting video bit rate is evaluated on a mobile HSDPA network with a physical layer bandwidth of 7.2 $Mbps$. The mobile client observes a base RTT of 95 ms to the streaming server. Apparently, TCP suffers from the large and variable RTT of the mobile network in that it significantly underutilizes the available bandwidth (Figure 6.13). The video rendering stalls frequently if the second adaptation set (2 $Mbps$ to 6 $Mbps$) is chosen since the TCP goodput is not continuously sufficient for the lowest quality level of 2 $Mbps$. Therefore, the TCP experiment is performed with the first adaptation set (0.8 $Mbps$ to 2.5 $Mbps$).

As PRRT achieves more throughput on the mobile network, the second adaptation set (2 $Mbps$ to 6 $Mbps$) is applied. Under the dynamic streaming application the protocol nearly continuously transmits a video bit rate of 5 $Mbps$, which is the second best quality level of the second adaptation set. The extensive low pass filtering of the delay-based congestion control scheme leads to a slower convergence of the bandwidth estimate at the beginning of the streaming session. However, within less than 50 s this turns into a highly stable throughput during the remaining session. According to Table 6.4, throughput and quality level are increased by more than 100 % on the mobile network.

6.3. Wireless Multicast

Due to the increasing number of portable and mobile consumer electronic devices, today's Internet paths include generally at least one wireless segment. Predominant architectures are IEEE 802.11 wireless as well as third generation (3G) and fourth generation (4G) mobile networks. Wireless and mobile networking architectures enable an ubiquitous Internet experience for portable and handheld devices. Mobile broadband standards, such as High-Speed Downlink Packet Access (HSDPA) and Long Term Evolution (LTE), are meanwhile providing an access bandwidth in the order of wired broadband Internet access, which makes them eligible as a replacement for wired connectivity in rural areas. In addition, the broadband network is commonly terminated by a wireless IEEE 802.11 access point in the home network. Obviously, Internet paths become increasingly heterogeneous and include in general at least one wireless segment.

The Internet Protocol hides the network path's heterogeneity from the upper layers such that the transport protocol is subject to a highly variable network state. Reliable transport protocols suffer from significant throughput variations on wireless networks, as they introduce physical packet corruption in addition to packet loss by queue overflow [28]. Therefore, the transport protocol becomes the performance bottleneck on wireless network paths as the physically available bandwidth is significantly underutilized. While the actual bandwidth provisioning on those networks is no longer the limiting factor for high definition media streaming, the quality of such services is severely affected by TCP's shortcomings on wireless paths.

The efficiency in wireless, large-scale media distribution is significantly improved via multicast, which introduces additional challenges in terms of adaptation to a representative network state for the whole group and scalable feedback handling. Specifically, the reliability of the feedback is a problem in shared wireless media as the number of collisions increases with the number of receivers such that the transport protocol has to find a good trade off between reliability, scalability and efficiency.

Table 6.5.: Spatial correlation in the wireless packet loss.

Receiver	1	2	3	4	5	6	#lost	#shared
1	-	9429	9814	8799	8633	9221	78633	43139
2	9429	-	11063	10114	8918	7337	103125	43310
3	9814	11063	-	10886	9635	7665	65663	54557
4	8799	10114	10886	-	9472	7745	64170	45372
5	8633	8918	9635	9472	-	9583	95578	44765
6	9221	7337	7665	7745	9583	-	59108	40150

6.3.1. Scalable Erasure Coding

Wireless IP standards implement partial reliability on MAC layer. Usually, an ARQ or Hybrid-ARQ scheme performs a limited number of transmission retries in order to reduce the number of packet erasures experienced on the transport layer. However, these wireless standards specify no error control at all or just open-loop error recovery in the multicast due to scalability issues in presence of receiver feedback. This design decision introduces mainly two problems. First, because of the absence of receiver feedback, the sender cannot switch the physical transfer mode consisting of modulation and coding schemes of different robustness. Second, without error control on MAC layer, the transport layer protocol is subject to highly variable packet loss rate.

Transport layer error control must be carefully designed in the multicast in order to preserve scalability. In the following, the scalability features of the PRRT protocol are motivated and evaluated via small multicast experiments leading to characteristic observations. The discussion includes the impact of unreliable feedback on PRRT's optimal configuration.

Spatial Correlation

Erasures that are in common among several receivers of the multicast group are called spatially correlated. Spatial correlation is either caused by interference near the concerned receivers or by protocol losses that affect the whole group. In particular, in the wireless multicast the spatial correlation of packet erasures is high since several closely located receivers might experience interference from the same source. Table 6.5 shows results from a long-term measurement of sending 10^7 multicast packets to 6 wireless receivers in an IEEE 802.11 wireless network. The results show that a receiver in the multicast scenario usually shares one packet erasure with at least one other receiver in more than 50% of all cases. On the other hand, for each pair of receivers generally about 10% of their lost packets are the same.

Basically, spatial correlation is beneficial for the multicast error coding as the same repair packet can replace lost packets at all affected receivers. In addition, the efficiency of block-erasure coding increases for closely located erasures at different receivers. This effect amortizes the redundancy information within the multicast group. However, as a further result, the feedback in reactive and adaptive error control tends to be synchronized among those receivers that are sharing the same packet losses, which leads to a flood of feedbacks at the same time slot. This is especially a problem on IEEE 802.11 wireless standards, where the medium is shared via a time division multiple access. Ex-

cessive feedback generates a large incoming network load at the sender and increases the probability of collisions on a shared wireless medium such that the error control itself might contribute to the overall packet loss process.

Inherent Scalability and Feedback Suppression

The predictably reliable protocol supports scalable error control in two ways: inherently via the reliability control and explicitly via the integrated feedback suppression. The efficiency model optimizes the protocol configuration under the objective of minimum coding overhead. However, in order to obtain a conservative parametrization, it assumes the worst case of spatially independent packet loss among the whole receiver group with an equal packet loss rate of GPLR (Section 5.2.2). As a result, protocol configurations with short coding blocks are penalized during the optimization as they require a larger amount of redundancy information under the assumed situation. Longer coding blocks amortize the repair packets better under spatially uncorrelated packet loss.

Figure 6.14 (top) shows that the optimization scheme increases the coding block length along with the packet loss probability and the number of receivers. The increase of the block size is quantized by the number of source packets becoming available within one request timer D_{REQ} such that successively reactive repair cycles are replaced by longer coding blocks. Correspondingly, more proactive repair packets are defined or larger reactive repair schedules are concentrated on the reduced number of repair cycles. Finally, the protocol configuration approaches an FEC configuration with a large block length and a purely proactive repair schedule, depending on the network's erasure rate and the size of the receiver group. This leads the coding overhead to tend asymptotically towards a constant value (Figure 6.14, bottom). Larger block sizes are preferred for higher source rates already under smaller group sizes and lower erasure rates. Figure 6.14 (bottom) shows also that the redundancy information amortizes better for higher source rates. Altogether, the increased proactivity in the protocol configuration along with the growing group size significantly decreases the probability of sending loss notifications at each receiver. Therefore, the reliability control implements inherent scalability via the objective of minimizing the redundancy information.

According to Section 4.2.2, the PRRT protocol implements timer-based feedback suppression, which is a common scheme for reliable or partially reliable multicast. However, the effects of timer-based feedback suppression are twofold in the predictably reliable error control. The protocol's time budget must incorporate the maximum suppression timer for each reactive repair cycle (Equation 4.10). This allows each receiver to set a random timer under this maximum in order to listen for foreign loss notifications and to discard own feedback if it is redundant. Figure 6.15 shows how this affects the number of feedbacks per second in a real scenario. The left figure is measured under a unicast setup with $P_e = 0.05$ and $RTT = 5\,ms$. The single receiver sends in average 20 loss notifications per second. The right figure shows the protocol performing under the same packet loss rate in a multicast group with 15 receivers. Feedback suppression with a maximum timer of $20\,ms$ is enabled during this experiment. Overall 2109 loss notifications are sent in the unicast setup, whereas 3751 loss notifications are sent in the multicast group. Obviously, the timer-based feedback suppression leads the number of feedback messages to scale significantly slower than the group size.

The second effect of the timer-based feedback suppression evolves from the time con-

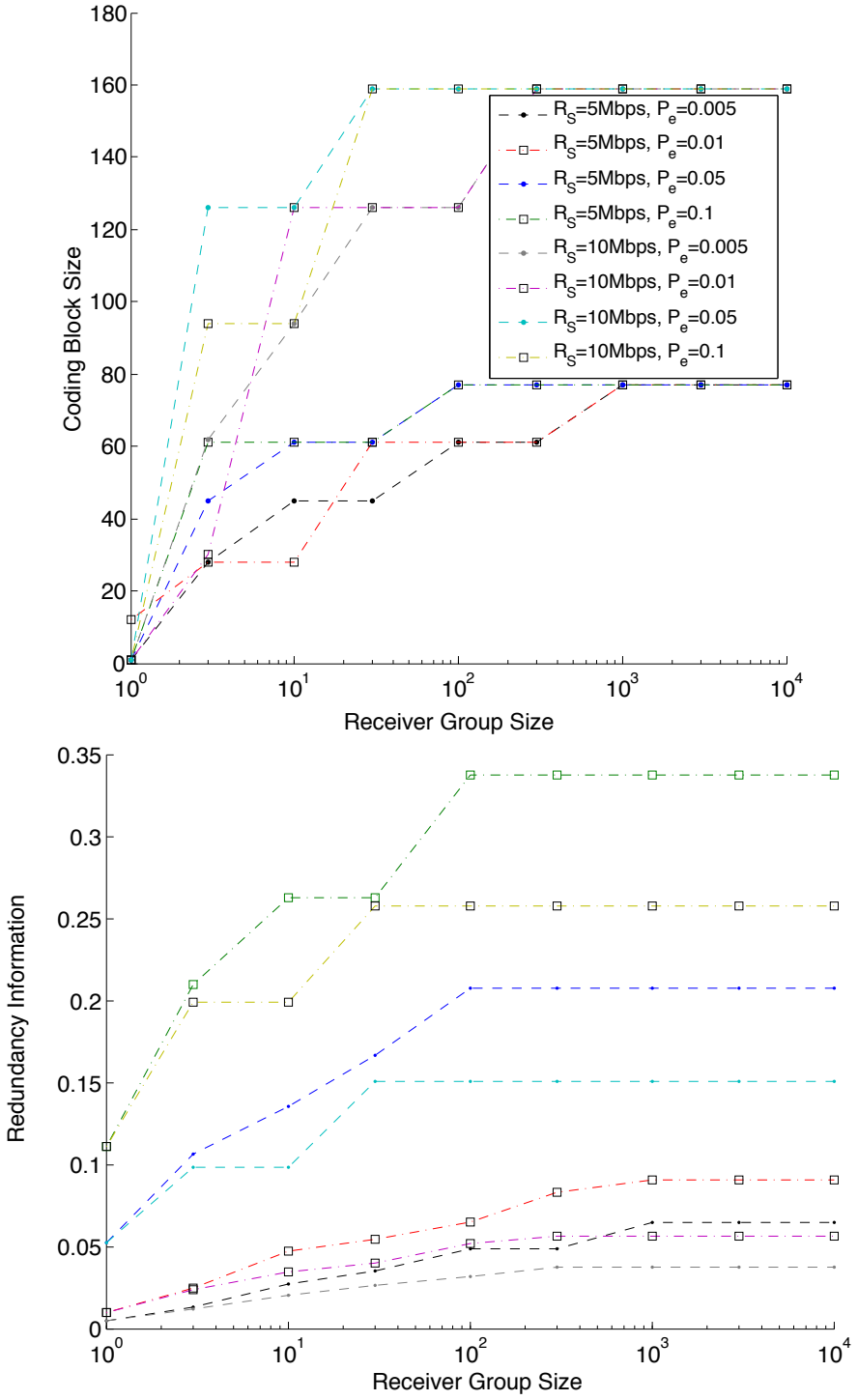


Figure 6.14.: Evolution of the coding block length k (top) and the redundancy information (bottom) with increasing receiver group size ($RTT = 5\,ms$, $D_T = 100\,ms$, $P_T = 10^{-6}$)

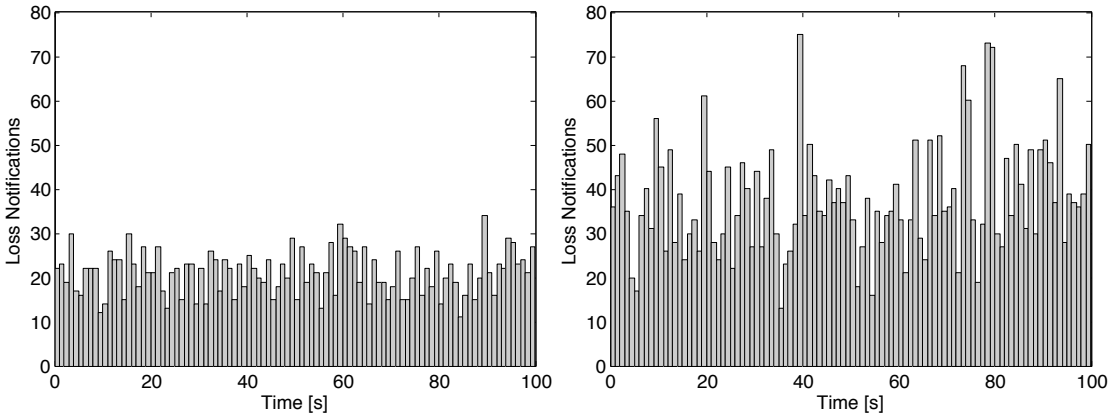


Figure 6.15.: Number of feedback packets per second (bin size $1\,s$) for one receiver (left) and 15 receivers (right) with a feedback suppression timer of $D_{SUP} = 20\,ms$ under a packet loss rate of $P_e = 0.05$ and source rate $S_R = 10\,Mbps$.

sumption by reserving the suppression timers in the overall time budget. As a result, PRRT's reliability control is additionally triggered to increase the proactivity in the protocol configuration since the number of reactive repair cycles is decreased under the reduced time budget. This effect is particularly visible in Table 6.6 for a maximum suppression timer of $D_{SUP} = 50\,ms$. For the higher packet loss rate this even leads the architecture to switch earlier to an adaptive FEC configuration for a group size of 10 receivers, which completely avoids the loss notifications.

Unreliable Feedback

The loss of feedback messages is particularly frequent on shared media such as the IEEE 802.11 wireless networks since downstream packets tend to collide with the loss notifications of the receivers. Unreliable delivery of loss notifications reduces the number of of repair cycles initiated at the sender such that the receiver observes an error floor higher than the desired residual packet loss rate (Section 4.2.3). As a response to the loss of reactive repair cycles, the reliability control algorithm increases the proactivity in the parametrization of PRRT.

According to Section 4.2.3, the impact of two factors is visible during the protocol optimization under unreliable feedback: Both the repetition of the last cycle's loss notification as well as the size of the receiver group compensate for the feedback loss rate. In order to evaluate these effects, the parameters of Table 6.7 are determined for a transmission scenario with $RTT = 5\,ms$ under a source rate of $R_S = 10Mbps$. The delay constraint is $D_T = 200\,ms$ and the reliability requirement defines a residual packet loss rate of less than $P_T = 10^{-6}$. Each receiver experiences an equal, i. i. d. packet loss rate of $P_e = 0.01$ and $P_e = 0.10$, respectively, which is equal to the feedback loss rate ($P_f = P_e$).

As a result of the unreliable feedback, the protocol prefers an adaptive FEC configuration with a long coding block for any group size under the packet loss rate of $P_e = 0.10$ (Table 6.7, left). For the lower loss rate, however, reactive protocol configurations still provide sufficient reliability such that purely reactive configurations are applied for group sizes of less than 5 receivers. For both loss rates, the impact of the unreliable feedback is

Table 6.6.: Effect of timer-based feedback suppression on the protocol configuration ($R_S = 10\,Mbps$, $RTT = 5\,ms$, $D_{RS} = 20\,ms$, $D_T = 200\,ms$).

N_R	P_e	$D_{SUP} = 0\,ms$				$D_{SUP} = 50\,ms$			
		k	N_P	P_r $\times 10^{-6}$	RI	k	N_P	P_r $\times 10^{-6}$	RI
1	0.01	1	[0, 1, 1, 1]	0.01	0.0101	1	[0, 1, 2]	0.01	0.0102
	0.10	1	[0, 1, 1, 1, 3]	0.10	0.1113	1	[0, 1, 5]	0.10	0.1500
3	0.01	30	[0, 1, 1, 1, 2]	0.22	0.0240	2	[0, 1, 2]	0.04	0.0302
	0.10	94	[7, 7, 15]	0.81	0.1992	80	[13, 14]	0.41	0.2007
5	0.01	94	[1, 2, 4]	0.71	0.0301	80	[2, 5]	0.27	0.0390
	0.10	62	[3, 3, 6, 11]	0.56	0.2172	80	[13, 14]	0.41	0.2215
10	0.01	126	[4, 4]	0.65	0.0348	80	[3, 4]	0.27	0.0422
	0.10	62	[3, 3, 6, 11]	0.56	0.2416	159	[41]	0.64	0.2579
15	0.01	126	[4, 4]	0.65	0.0363	80	[3, 4]	0.27	0.0444
	0.10	159	[41]	0.64	0.2579	159	[41]	0.64	0.2579

Table 6.7.: Effect of feedback repetition in the last repair cycle ($R_S = 10\,Mbps$, $RTT = 5\,ms$, $D_{RS} = 20\,ms$, $D_T = 200\,ms$).

N_R	P_e (P_f)	$d = 1$				$d = 3$			
		k	N_P	P_r $\times 10^{-6}$	RI	k	N_P	P_r $\times 10^{-6}$	RI
1	0.01	1	[0, 1, 1, 1]	0.04	0.0101	1	[0, 1, 1, 1]	0.01	0.0101
	0.10	159	[41]	0.64	0.2579	1	[0, 1, 1, 1, 3]	0.47	0.1113
3	0.01	30	[0, 1, 1, 1, 2]	0.65	0.0240	30	[0, 1, 1, 1, 2]	0.22	0.0240
	0.10	159	[41]	0.64	0.2579	94	[7, 7, 15]	0.81	0.1992
5	0.01	126	[3, 5]	0.65	0.0313	94	[1, 2, 4]	0.71	0.0301
	0.10	159	[41]	0.64	0.2579	62	[3, 3, 6, 11]	0.56	0.2172
10	0.01	126	[4, 4]	0.65	0.0348	126	[4, 4]	0.65	0.0348
	0.10	159	[41]	0.56	0.2579	62	[3, 3, 6, 11]	0.56	0.2416
15	0.01	126	[4, 4]	0.65	0.0348	126	[4, 4]	0.65	0.0348
	0.10	159	[41]	0.64	0.2579	159	[41]	0.64	0.2579

not visible for group sizes beyond 10 receivers (compare left parameter set in Table 6.7 with left parameter set in Table 6.6). At this point the redundant feedback of the multicast group compensates for the erased loss notifications. As the protocol prefers purely proactive configurations for large receiver groups under higher loss rates, the success of the error control does not depend on the reliable transmission of loss notifications such that the effect of the feedback loss vanishes.

The right parameter set in Table 6.7 shows the optimum protocol configurations under the assumption that the loss notification is repeated $d = 3$ times in the final repair cycle. The resulting parameter set is identical to the left parameter set in Table 6.6, which is obtained under reliable feedback delivery. Obviously, the repetition of the last cycle's feedback completely eliminates the effect of unreliable feedback in the addressed scenario.

6.3.2. Wireless Media Distribution

The IEEE 802.11 family of wireless network standards offers a convenient and inexpensive way to deliver media streams within the home network area. The infrastructure is equally attractive for the media distribution in public hot spot environments. Particularly for the local streaming of multimedia data, the IEEE 802.11 standards have a significant place in the networking standards of consumer electronic devices such as the DLNA[10] interoperability guidelines. Recent consumer electronic devices are generally equipped with a corresponding wireless network interface.

Wireless media multicast is notably beneficial for the distribution of broadcast TV within home networks. The TV signals might be received from a conventional broadcast network or via an IPTV service. Such services tend to be consumed in parallel by several receiver devices in home and public areas such that multicast provides a significant efficiency gain compared to multiple unicast sessions. Besides, the bandwidth provisioning on the wireless network is usually not sufficient to carry the load of multiple TV channels with high quality.

IPTV services rely on multicast distribution on the delivery network in order to serve millions of subscribers. Without adequate reliability on the transport layer, those services must be terminated by wired devices since IEEE 802.11 multicast neither applies error control in the MAC layer nor rate adaptation for different robustness. In the following, PRRT's performance is evaluated under the distribution of live broadcast services on wireless local networks.

Related Work in wireless Video Distribution

Substantial research has been done on improving the transport of real-time media on wireless networks. On transport layer the proposed solutions concentrate on extensions to TCP's error and congestion control. These approaches implement loss tolerance (LT-TCP [150]) or loss differentiation (TCP-Westwood [101]) In order to increase TCP's throughput under high loss rates. LT-TCP wraps adaptive FEC proactively and reactively into TCP-SACK (Selective Acknowledgment) and achieves a goodput of $4\,Mbit/s$ at a packet loss rate of 20% by adding roughly 50% overhead. TCP-Westwood in its Buffer and Bandwidth Estimation (BBE) variant is able to double the throughput of unmodified TCP-SACK under high loss rates. Being based on TCP, those schemes, however, do not support multicast distribution.

In the IEEE 802.11 MAC layer, so called leader-based protocols achieve high correction performance under low delay and overhead [95]. It is recommended to combine them with a transport or application layer scheme in order to deal with a residual error floor. VPAL implements a dedicated packet adaptation layer for reliable video multicast [35]. The authors apply Raptor codes on frame level in IEEE 802.11n wireless networks. Raptor codes cause significant coding overhead under low delay constraints since the number of required excess symbols amortizes only for huge block lengths. A residual loss rate of 10^{-3} under 10% packet loss rate is achieved with an overhead of 28%, which significantly exceeds the overhead added by an optimized HEC.

[10] Digital Living Network Alliance, www.dlna.org

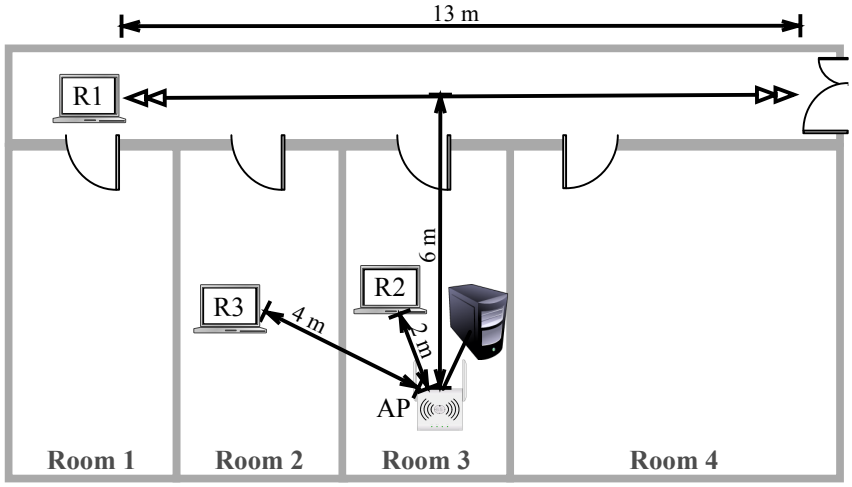


Figure 6.16.: Experimental setup.

Experimental Setup

The application scenario consists in the local redistribution of digital satellite TV via PRRT on an IEEE 802.11a wireless network. During the following experiments a standard definition TV channel with an average source rate of 4 $Mbps$ as well as a high definition channel with average source rate 12 $Mbps$ are delivered via multicast. The wireless stations are associated with the access point, whereas the sender application runs on a wired host with a connection to the digital broadcast source. The AP is fixed to the 18 $Mbit/s$ transmission mode for the standard definition stream and to 36 $Mbit/s$ for the high definition stream, respectively. In the experiments, the PRRT protocol is configured to achieve a residual packet loss rate of $P_T = 10^{-6}$ under a delay constraint of $D_T = 200\,ms$ while the update interval of the reliability control is set to $T_{UPD} = 200\,ms$ and network state feedback is sent with a period of $T_{NFB} = 100\,ms$. The protocol compensates for host latencies with $D_{RS} = 20\,ms$. Each PRRT datagram carries seven MPEG-2 Transport Stream packets [53], which results in a payload size of 1316 $byte$.

Predictable Reliability in the wireless Multicast

First, the standard definition TV stream is sent to one mobile (R1) and one stationary receiver (R2) via multicast (Figure 6.16). The mobile receiver moves periodically back and forth on the corridor. Therefore, it perceives different obstruction scenarios, i. e. a varying number of walls between the AP and the wireless station. The signal strength is reduced by roughly 10 dB for each wall in the line of sight, which results in an oscillating signal-to-noise ratio (SNR) at the mobile receiver (Figure 6.17). High peaks in the measured PLR correlate with minimum values in the SNR. PRRT's error control corrects the occurring peak loss rates of nearly 20 % down to a cumulative residual packet loss rate of less than $5 \cdot 10^{-5}$ for the mobile receiver. The stationary receiver does not experience residual packet losses. During the whole experiment, the adaptive error control adds less than 2 % of redundancy information. It is visible that the graph of the redundancy information follows both the increasing PLR and the increasing RTT, which indicates

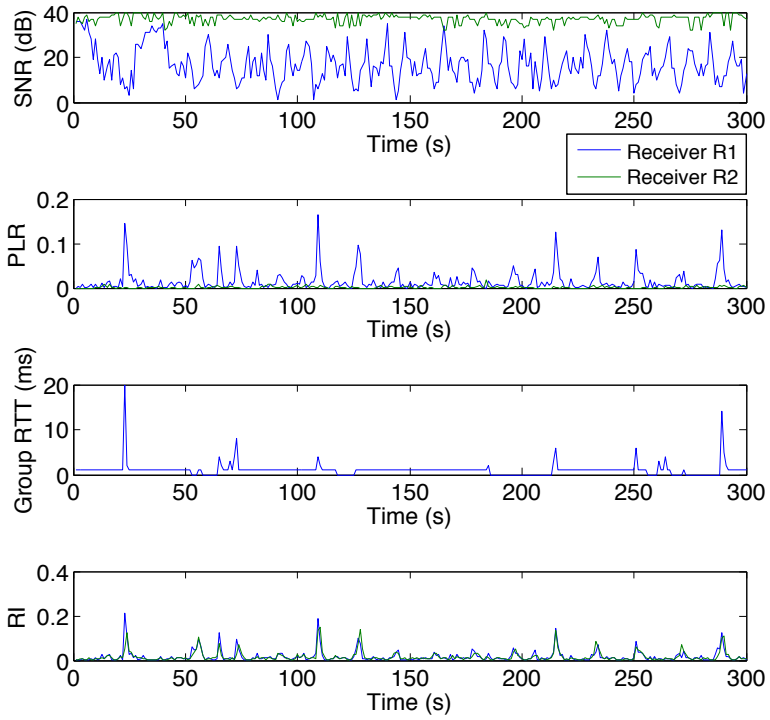


Figure 6.17.: Transmission of a 4 $Mbit/s$ standard definition TV stream to a mobile receiver R1 and a stationary receiver R2.

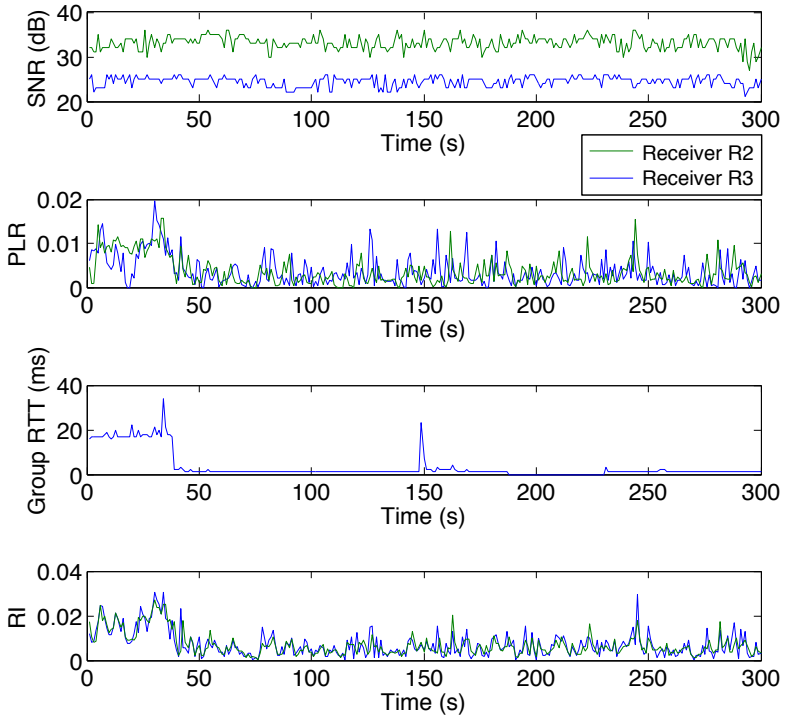


Figure 6.18.: Transmission of a 12 $Mbit/s$ high definition TV stream to two stationary receivers R2 and R3.

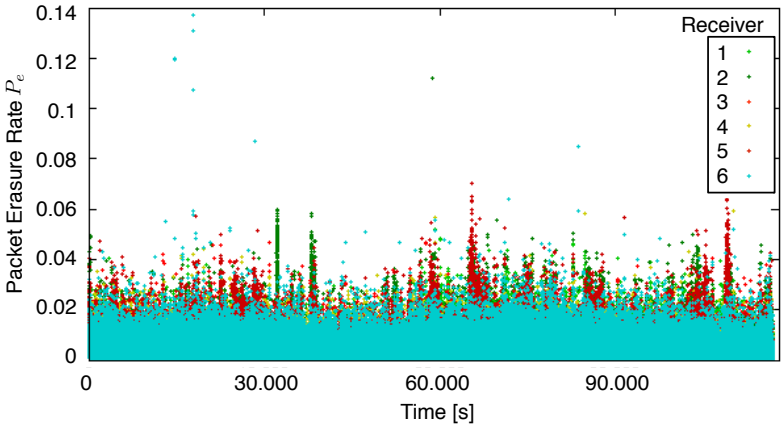


Figure 6.19.: Packet loss rate during a long-term experiment.

Table 6.8.: PRRT's multicast performance for 6 wireless receivers. P_e and P_r are long-term average values obtained after sending $5 \cdot 10^7$ packets.

Receiver	1	2	3	4	5	6
$\bar{P}_e$	$3.33 \cdot 10^{-3}$	$3.75 \cdot 10^{-3}$	$3.01 \cdot 10^{-3}$	$3.04 \cdot 10^{-3}$	$3.91 \cdot 10^{-3}$	$3.13 \cdot 10^{-3}$
P_r	$0.02 \cdot 10^{-6}$	$2.34 \cdot 10^{-6}$	0	$0.84 \cdot 10^{-6}$	$3.74 \cdot 10^{-6}$	$0.58 \cdot 10^{-6}$

the adaptivity of the transport protocol.

In a second experiment the high definition service is multicasted to two stationary receivers (R2 and R3, Figure 6.16). The signal attenuation due to the wall is visible in the significantly lower SNR at receiver R3 (Figure 6.18). R2 observes a lower signal quality as well, which is a result of the less robust physical transmission mode that is selected for the high definition stream. The lower SNR causes a higher PLR at both receivers with large variations. Due to the adaptive error control, however, both receivers experience a residual packet loss rate of less than 10^{-5}.

The third experiment performs a long-term evaluation of the protocol's correction performance in the wireless multicast of a 4 *Mbps* stream to six stationary receiver nodes. Table 6.8 presents the results after sending $5 \cdot 10^7$ PRRT packets of which overall 370147 packets are lost by the receiver group. All receivers experience in long-term an average packet loss rate of roughly $4 \cdot 10^{-3}$. However, the peak packet loss rate exceeds sporadically 10 % (Figure 6.19). The results of the long-term experiment confirms that PRRT achieves the reliability requirement of $D_T = 10^{-6}$ with high probability.

7. Conclusion

Among the manyfold applications implemented on top of the Internet Protocol, the timely and high quality multimedia distribution remains a challenging topic. The available thesis fundamentally discusses the feasibility of such low latency and high data rate streaming services on unmanaged Internet infrastructures. The inelastic nature of multimedia streams introduces several new objectives into the design of transport layer protocols. Moreover, those objectives are not mutually independent. Specifically this refers to adequate reliability, limited delay and a minimum bandwidth requirement. All these values are implicitly connected and limited by Shannon's theorem on noisy channel coding.

Today's media applications must choose between a managed end-to-end path or alternatively, a totally or partially reliable transport protocol in order to deliver streaming content between networked endpoints. Whereas the managed Internet ensures the provisioning of the application's QoS requirements, it relies on the support of the underlying infrastructure. The transport protocols, on the other hand, stand out with their capabilities of self-management and convenient deployment. However, they significantly affect the quality of the media distribution under variable and dynamic network conditions. A fundamental goal of this thesis is therefore the evaluation of an intermediate path between both concepts: the design of a self-managing Internet transport layer for continuous real-time packet streams that makes the underlying network dynamics transparent under the specific constraints of multimedia applications.

7.1. Summary

The observations obtained during the design and the evaluation of the protocol architecture for predictably reliable media transport within this thesis converge into three essential conclusions:

- The transport protocol must provide a high degree of flexibility in order to customize the error control in fine-grained steps to the properties and constraints of networks and applications.

- The communication of QoS requirements to the transport layer and the propagation of goodput estimates to higher layers is crucial. Even though this form of communication conflicts with the strict ideas of the Internet layering model, the exchange of cross-layer information provides the most promising gains in network throughput within numerous recent networking architectures.

- As today's Internet paths are heterogeneously wired and wireless with highly dynamic channel capacity, the transport protocol must continuously follow the network state in order to instantly adapt the transport parameters and to verify their feasibility.

The transport architecture presented within this thesis results from extensive observation and evaluation of the dynamics of real-time media transport on various Internet infrastructures. In particular, it contributes the following components.

Predictable Reliability

Due to the asynchronous nature of packet-switching networks, the delivery of delay-sensitive services along Internet paths is problematic if a certain level of reliability must be offered. As opposed to partially reliable protocols that are not able to maintain the desired reliability under a delay constraint, this thesis introduces the term *predictable reliability*. By dynamically trading redundancy for delay, a predictably reliable protocol can achieve a specific reliability level under a given time constraint with high probability. This is a result of the joint modeling of network state and protocol performance, whereas the protocol itself supports fine-grained parametrization under proactive and reactive error control.

Real-time Error Control

Under predictably reliable error control, the timely delivery of each transmission unit has highest priority. Therefore, each packet is assigned a strict delivery deadline after which it must be available to the receiving application, otherwise it is considered lost. In order to meet this requirement, the presented predictably reliable protocol implements two essential features. First, all receiving nodes are synchronized to the sender such that they share the same time base. Second, depending on this time base, all protocol actions such as error coding and feedback are scheduled via explicit timers. In contrast to an event-driven protocol state machine, those actions run in parallel to the processing of the source packet stream such that the continuous packet flow is not interrupted by the error control. These properties make the transport architecture particularly suitable for interactive applications and streaming services with a specific delay constraint.

Self-managing Reliability Control

The protocol layer allows for dynamic and flexible configuration. Given the suitable modeling of the network state, it is self-managing in that it observes the characteristics of the underlying network infrastructure such as packet loss, burstiness and delay. On top of the measured and modeled network state, the protocol performance model enables the optimization of the protocol parametrization under various objectives and constraints, depending on the usage scenario and the underlying infrastructure. Flexibility and self-management under the given constraints make the protocol a valuable basis for upper-layer and overlay data communications schemes. As it provides a transport pipe with constant delay and controllable reliability, it conveniently supports multi-hop error control under delay and reliability constraints.

Flexible Protocol Architecture

The predictably reliable protocol architecture as well as the practical implementation into the PRRT protocol provide a modular and extensible environment. By implementing proactive and reactive packet repair mechanisms and specifying the related parameter signaling, the protocol supports a wide range of packet-level error coding schemes.

The explicit scheduling of feedback packets allows for the implementation of different acknowledgment policies with different scalability characteristics in the multicast.

The single modules of the protocol architecture implement thin and well-defined interfaces, which ensures a convenient replacement of single modules without affecting the remaining components. For instance, the block-erasure model might be replaced by a more sophisticated model for a specific network infrastructure. Similarly, the packet-level coding unit may be substituted by a another block coding scheme along with an appropriate update of the protocol performance model. Furthermore, the rate estimate, which contributes an essential constraint to the protocol optimization, might be obtained from several alternative bandwidth estimation or rate-based congestion control techniques.

7.2. Future Work

Predictably reliable media transport is tangent to various research topics. Future research interests are particularly motivated in the field of the protocol parameter optimization, the reliability control, the congestion control as well as the application of the protocol in upper layer architectures and overlay schemes. Potential extensions to these modules are discussed in the following.

Protocol Parameter Optimization

The protocol performance model as well as the parameter optimization algorithm are optimized under the objective of limited complexity in order to enable the instant adaptation of protocol parameters in real implementations. However, the computational effort at the sender, which solves the actual optimization problem, might still be reduced. A potential approach would be the implementation of a learning algorithm that either successively reduces the result space of potential protocol parameters or remembers characteristic settings for certain parameters under the corresponding network state. Moreover, the parameter search might be implemented progressively. Starting from few parameter sets that are known to be sufficiently reliable but suboptimal under the current situation, the search would incrementally approach the optimum solution. Pre-calculated parameter tables might be generated for specific scenarios with reasonably limited input parameter space or result space.

Reliability Control

In order to adaptively catch a wider range of disturbances in the network state, the reliability control might incorporate the measured residual packet loss rate into the adaptation process. For instance, reliability feedback would allow for the adaptive tuning of the block-erasure model's sample size and thereby increase the margin of error (Section 5.2.1) on the estimated mean. This results in a more conservative estimate of the network path's packet loss rate. Open issues with respect to such a setup are in particular the formulation of a good estimator for the residual packet loss rate in long-term as well as a compensation for the delayed reliability feedback in case of large round trip times.

Congestion Control

The protocol implementation presented within this thesis is prepared for various instances of equation-based congestion control. Section 6.2 contributes results that are obtained under the application of a delay-based control equation in a unicast setup. Bandwidth and rate are concave metrics and their negotiation in the multicast is still a challenging problem under dynamic channel capacity. Besides the optimization of congestion control in the multicast, the suitable scheduling of media frames must be studied under the variable bandwidth estimate. Dynamic video streaming is currently a promising architecture to introduce a quantized elasticity into the multimedia streaming. For inelastic services, the protocol might be combined with appropriate admission control schemes.

Upper Layer Architectures

As a feature of the predictable reliability under a given delay constraint, the presented architecture is a valuable basis for a variety of overlay schemes in meshed networks. In such schemes the end-to-end QoS constraints are distributed among the individual segments of the network path according to a reasonable splitting of the end-to-end path into separate loss domains. The protocol is then optimized under the partial constraints in a multi-hop transmission scheme. Similarly, the protocol might serve as an elementary transport component for multi-path, path diversity and load balancing schemes. For all those schemes it is crucial to appropriately gather knowledge about the network topology and to formulate the corresponding upper-layer optimization problem of distributing the QoS constraints among multiple segments and multiple paths.

A. Appendix: Related Publications

1. Gorius, M.; Herfet, Th.: *Predictable Reliability for Media Streaming over unmanaged Internet*, accepted at the NEM Summit, Istanbul, October 2012

2. Gorius, M.; Petrovic, G.; Herfet, Th.: *Efficient and Low-Delay Error Control for Large-BDP Networks*, Elsevier International Journal for the Computer and Telecommunications Industry, 2012

3. Gorius, M.; Shuai, Y.; Herfet, Th.: *Dynamic Media Streaming over wireless and mobile IP Networks*, IEEE International Conference on Consumer Electronics - Berlin (ICCE-Berlin), Berlin, September 2012

4. Gorius, M.; Shuai, Y.; Herfet, Th.: *Dynamic Media Streaming under Predictable Reliability*, IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB), Seoul, June 2012

5. Gorius, M.; Shuai, Y.; Herfet, Th.: *Predictably Reliable Media Transport over Wireless Home Networks*, IEEE Consumer Communications and Networking Conference (CCNC), Las Vegas, January 2012 – *Best Student Paper Award*

6. Gorius, M.; Petrovic, G.; Herfet, Th.: *Predictably Reliable Real-time Transport over Large Bandwidth-Delay Product Networks*, NEM Summit, Torino, September 2011

7. Gorius, M.; Miroll, J.; Karl, M.; Herfet, Th.: *Predictable Reliability and Packet Loss Domain Separation for IP Media Delivery*, IEEE International Conference on Communications (ICC), Kyoto, June 2011

8. Gorius, M.; Miroll, J.; Herfet, Th.: *Service compatible efficient 3D-HDTV Delivery*, Proceedings of the 14th ITG Conference on Electronic Media Technology, Dortmund, March 2011

9. Karl, M; Gorius, M; Herfet, Th.: *A Distributed Multilink Media Transport Approach*, International Conference on Intelligent Network and Computing (ICINC), Kuala Lumpur, November 2010

10. Karl, M; Gorius, M; Herfet, Th.: *Routing: Why less intelligence is sometimes more clever*, IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB), Shanghai, March 2010

11. Gorius, M.; Karl, M.; Herfet, Th.: *Optimized Link-level Error Correction for Internet Media Transmission*, NEM-Summit, St. Malo, September 2009

12. Gorius, M.; Herfet, Th.: *Echtzeit-Fehlerschutz für die drahtlose paketbasierte Fernsehübertragung*, 23rd annual conference of the FKTG, Munich, May 2008

13. Tan, Guoping; Herfet, Th.; Gorius, M.: *Evaluation of the Performance of a Hybrid Error Correction Scheme for DVB Services over IEEE 802.11a*, IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB), Orlando, March 2007

14. Herfet, Th.; Gorius, M.: *Anwendung eines RTP-basierten Hybrid-Fehlerschutzes in der kabellosen IP-basierten Fernseh-Übertragung*, 12th ITG Conference on Electronic Media Technology, Dortmund, March 2007

15. Herfet, Th.; Gorius, M.: *IP-basierte Verteilung von DVB-Inhalten in lokalen Netzwerken,* FKT - Fachzeitschrift für Fernsehen, Film und elektronische Medien, pp. 112-118, March 2007

B. Appendix: Protocol Implementation

In the course of this thesis the predictably reliable real-time protocol has been implemented into a runtime-loadable Linux kernel module[1]. The developed protocol stack fulfills the protocol specification defined in Chapter 2. The protocol implementation, conveniently integrates predictable reliability into networked applications as it wraps a standard BSD socket interface with few protocol-specific modifications and extensions. The source package includes small command line tools that implement PRRT's main features and allow for special configuration via command line options. Given a real-time media source, the example tools conveniently set up a networked pipe that transmits the source data between remote network sites with predictable reliability.

B.1. Protocol Interface

In addition to the host address and the source or destination port, which both define the Transport Service Access Point (TSAP) of a transport layer protocol, PRRT requires the exchange of additional information in order to enable the bidirectional error control. As the protocol itself is connectionless, session initiation must be implemented on higher layers in order to communicate the host addresses and ports for source data and feedback transmission. In order to provide the required communication and functionality, a C++ Socket library maps the PRRT socket API to the related system calls of the BSD socket interface and registers the required components in the PRRT protocol stack. The protocol stack implements the predictably reliable error control, the reliability control and the delay-based congestion control in a multi-threaded environment such that ordinary *send()* and *receive()* calls at the PRRT hosts entirely abstract the complex protocol functionality. As soon as a character buffer is written into the sender socket via *send()*, the protocol stack determines the delivery deadline of the datagram and initiates the error control operations.

the PRRT socket functionality that is implemented via a multi-threaded architecture within the network stack. receive() obtains a packet not earlier than packet timeout such that error control can is completed transparently.

use of connect and bind slightly modified since each PRRT session is bidirectional as it requires a feedback channel.

The protocol's programmer interface is defined in a C++ socket library as follows:

- **prrt_socket()** instantiates a new PRRT socket and allocates the required system resources. The socket is registered in a global register file ... the state of all PRRT sockets. For each PRRT socket, the library creates an entry in the Linux */proc* file system that returns information and statistics about the socket state (current protocol parameters, number of source and repair packets sent/received),

[1] Source code available for download under http://www.nt.uni-saarland.de/en/projects/running-projects/prrt.html

the observed network state (PLR, RTT, CC) and the protocol performance (P_r, RI).

- **connect()** sets up a sending PRRT socket. This function configures the *destination address*, the *destination port* as well as the *feedback address*. *destination address* and *destination port* might identify either a unicast receiver or a multicast group. *feedback address* identifies the local interface PRRT must listen at in order to obtain receiver feedback. In order to benefit from feedback suppression in multicast, *feedback address* should represent a multicast group. The feedback port is automatically set to *destination port + 1*. The function returns successfully if the required addresses and port numbers are valid and the automatically assigned feedback port is not already in use.

- **bind()** sets up a receiving PRRT socket. This function configures the *source address*, the *source port* as well as the *feedback address*. *source address* and *source port* might identify either a local network interface of the receiver or a multicast group. *feedback address* identifies the local interface PRRT must listen at in order to obtain receiver feedback. In order to benefit from feedback suppression in multicast, *feedback address* should represent a multicast group. The feedback port is automatically set to *destination port + 1*. The function returns successfully if the required addresses and port numbers are valid and the automatically assigned feedback port is not already in use.

- **close()** closes the PRRT socket and de-allocates all system resources. The socket is being deleted from the register file and the related entry in the */proc* file system is being removed.

- **send()** wraps a buffer of characters into a datagram of the required length and sends it to the receiver socket.

- **recv()** receives a datagram of specific length from the sender socket and returns the encapsulated buffer of characters.

- **setsockopt()** configure the PRRT socket via general UDP socket options and special PRRT socket options such as delay constraint, reliability constraint, rate constraint and manual settings for coding and protocol parameters.

- **getsockopt()** obtain the current settings for general UDP socket options and special PRRT socket options.

Additional socket functions beyond the BSD interface

- **getstats()** returns current socket statistics such as the socket state (current protocol parameters, number of source and repair packets sent/received), the observed network state (PLR, RTT, CC) and the protocol performance (P_r, RI).

- **getstreamrate()** returns the current goodput estimate for which the application's reliability requirement is satisfiable. If the application feeds data into the sender socket at a rate beyond the obtained goodput estimate, it might experience increased residual packet loss rate as PRRT's packet scheduler masks away excessive source and repair packets.

B.2. Example Tools

The following example tools implement a pair of simple PRRT sender and receiver. The tools presented in the following have been applied in order to obtain the experimental results in Chapter 6. They provide a simple self-test environment by transmitting an artificial character source but also a networked pipe functionality between the endpoints of a one-to-one or one-to-many real-time media streaming application. The tools export special protocol options such as fixed configuration or activation and deactivation of single protocol features.

Sender "prrtcat"

prrtcat is a PRRT-based sender application that implements connection-less streaming of real-time source data to a unicast PRRT receiver or a multicast group of PRRT receivers. Name and functionality are adopted from the POSIX tool netcat[2], which reads character data from the standard input and sends it to a remote socket via UDP or TCP. Similarly, prrtcat reads data from the standard input and encapsulates them into PRRT packets while performing the predictably reliable error control. Alternatively, the tool sends out an artificial character stream with a specified real-time source rate. This option is particularly interesting for a self-test or a performance measurement under evaluation of the socket statistics at the receiver.

The software is configured via the following command line options:

```
Sender options:
  -sp <port>                 destination port [default: 5004]
  -suc <unicast address>     unicast destination address
  -smc <multicast address>   multicast destination address
  -fb <address>              feedback address
  -code <coding parameters>  [default: 200,2,20,40,0,1,0,1,2]
                               i.e.: D_T(delay constraint)=200ms, D_C=2ms, D_FEC=20ms,
                               D_REQ=40ms, D_FB=0ms, k=1, N_P=[0 1 2]
  -g <target PLR>            [default: 0.000001]
  -j <data rate>              Sends from an artificial data source with specified data
                               rate (in Kbps) instead of reading from stdin (default)
  -ad <0|1|2>                Enable adaptation [default: 0]
                               0: no adaptation
                               1: table lookup
                               2: online optimization
  -i <update interval>        update interval of the parameter adaptation in
                               milliseconds [default: 200]
```

Receiver "prrtrecv"

prrtrecv offers basic PRRT receiver functionality. The tool instantiates a PRRT receiver socket and writes the received character buffers to the standard output. Except for the congestion control and the feedback scheduling, the PRRT receiver is a purely passive component that obtains all parameter settings via the corresponding protocol headers. Therefore, prrtrecv does not provide any options for coding parameters and application constraints. Bandwidth estimation is either provided by TFRC or a delay-based congestion control equation derived from FAST TCP [160].

```
Receiver options:
  -rp <port>                 destination port (listen) [default: 5004]
```

[2] http://linux.die.net/man/1/nc

```
-ruc <unicast address>      unicast destination address (listen)
-rmc <multicast address>    multicast destination address (listen)
-bw <0|1|2>                 enable bandwidth estimation [default: 0]
                              0: no bandwidth estimation
                              1: TCP-friendly rate control (TFRC)
                              2: delay-based congestion control
-i <updateInterval>         update interval of the bandwidth estimation in
                              milliseconds [default: 200]
-fs <D_SUP>                 enable feedback suppression with maximum suppression
                              timer D_SUP [default: 0]
```

Example Usage

The following command lines provide some typical example usages of the prrtcat and prrtrecv:

- Send from the artificial character source with a source rate of 5 $Mbps$ to the multicast group 224.0.1.2:5004 while specifying a delay constraint of 400 ms and fixed coding parameters with $k = 1$ and $N_P = [0, 1, 1, 4]$:

  ```
  $> prrtcat -j 5000 -code 400,1,20,40,0,1,0,1,1,4 -suc 224.0.1.2 -fb 224.0.1.2
  ```

- Set up a multicast receiver joining and listening to the group 224.0.1.2:5004 (character stream from standard output might be redirected to /dev/null) and sending feedback with a maximum suppression timer of 50 ms:

  ```
  $> prrtrecv -fs 50 -rmc 224.0.1.2 224.0.1.2 [> /dev/null]
  ```

- Send from the artificial character source at sender 192.168.0.1 with adaptive source rate to the unicast receiver 192.168.0.17:5004 under a delay constraint of 400 ms and enable online parameter adaptation with update interval 200 ms:

  ```
  $> prrtcat -j 0 -code 400,1,20,40,0,1,0,1,1,4 -ad 2 -i 200 -suc 192.168.0.17
     -fb 192.168.0.1
  ```

- Set up the corresponding unicast receiver 192.168.0.17 sending feedback to 192.168.0.1 and returning a bandwidth estimate every 100 ms

  ```
  $> prrtrecv -bw 2 -i 100 -ruc 192.168.0.17 192.168.0.1 [> /dev/null]
  ```

- Set up an online-adaptive networked pipe with predictable reliability between *streaming_server* at the sender 192.168.0.1 and *media_renderer* at the receiver 192.168.0.17. *streaming_server* writes data to standard output and *media_renderer* reads from standard input.

  ```
  $> media_server | prrtcat -code 400,1,20,40,0,1,0,1,1,4 -ad 2
     -i 200 -suc 192.168.0.17 -fb 192.168.0.1

  $> prrtrecv -ruc 192.168.0.17 192.168.0.1 | media_renderer
  ```

- Obtain current socket state and protocol statistics from the */proc* file system of an PRRT socket identified via *socket_number*:

  ```
  $> cat /proc/prrt/<socket_number>/stats
  ```

C. Appendix: Protocol Performance Analysis

The protocol performance analysis is implemented into the PRRT socket library as well as into a stand-alone tool for off-line evaluation of protocol parameters. The tool creates the timing model and the block-erasure model according to the network state parameters defined via command line arguments. It calculates the block error distribution for the Bernoulli model if the burstiness coefficient is zero, otherwise it samples the distribution from the simplified Gilbert-Elliott model via the combinatorial closed-form algorithm from Section 3.3.1. The tool is efficiently implemented in the C programming language under application of dynamic programming schemes in order to obtain fast solutions for several sub-problems of the parameter search algorithm. The software implements the greedy search algorithm from Section 5.1.2.

C.1. Implementation of the Performance Model

The parameter search tool obtains the application constraints and the network state parameters via the command line interface. Since the tool operates in stand-alone mode, additional application parameters such as channel and source bandwidth as well as the number of receivers must be provided. The signature also contains the manual protocol parameters.

```
$> optimize_prrt <Target Delay [ms]> <Target Reliability>
<Channel Bandwidth [bps]> <Source Bandwidth [bps]>
<Number of Receivers> <Packet Loss Rate> <Round Trip Delay [ms]>
<Feedback Loss Rate> <Burstiness Coefficient> <Payload Size [bytes]>
<Host Delay [ms]> <Transmission Delay [ms]> <Error Detection Delay [ms]
<Feedback Delay [ms]> <Feedback Suppression Delay [ms]>
```

The following listing shows an example run for the optimization of protocol parameters under a delay requirement of $300\,ms$ and a reliability constraint of 10^{-5} for a media stream with source rate $5\,Mbps$ and payload size $1316\,byte$ sent over a network with $10\,Mbps$ available bandwidth to a single receiver. The network is characterized by an i. i. d. packet loss rate of $10\,\%$ and a round trip delay of $50\,ms$, whereas the feedback channel is reliable. $20\,ms$ are reserved for host latencies and $10\,ms$ for the transmission of each repair packet schedule. Error detection is assumed to be performed within $8\,ms$.

```
$> optimize_prrt 300 0.00001 10000000 5000000 1 0.1 50 0.0 0.3 1316 20 10 8 0 0

Input:
Target Delay [ms]: 300.000000
Target Reliability: 0.000010
Channel Bandwidth [bps]: 10000000.000000
Source Bandwidth [bps]: 5000000.000000
Number of Receivers: 1
Packet Loss Rate: 0.100000
Round Trip Delay [ms]: 50.000000
Feedback Loss Rate: 0.000000
Burstiness Coefficient: 0.300000
Payload Size [bytes]: 1316
Host Delay [ms]: 20.000000
Transmission Delay [ms]: 10.000000
Error Detection Delay [ms]: 8.000000
Feedback Delay [ms]: 0.000000
Feedback Suppression Delay [ms]: 0.000000

Results:
D_t [ms]: 300.000000
P_r: 0.000001
RI: 0.113000
D_C [ms]: 285.105591
D_FEC [ms]: 45.105598
D_REQ [ms]:
k: 1
N_C: 3
Np: [0 1 1 3]
```

Bibliography

[1] 3GPP. 3GPP TS 22.146 v10.0.0 - Multimedia Broadcast/Multicast Service (MBMS); Stage 1, 2011.

[2] L. Abeni, A. Goel, C. Krasic, J. Snow, and J. Walpole. A measurement-based analysis of the real-time performance of linux. In *Proceedings of the Eighth IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS) 2002*, September 2002.

[3] B. Adamson, C. Bormann, M. Handley, and J. Macker. Negative-acknowledgment (NACK)-Oriented Reliable Multicast (NORM) Protocol. RFC 3940 (Experimental), November 2004.

[4] J. S. Ahn, P. B. Danzig, Z. Liu, and L. Yan. Evaluation of TCP Vegas: emulation and experiment. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM) 1995*, pages 185–195, August 1995.

[5] S. Akhshabi, A. C. Begen, and C. Dovrolis. An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP. In *Proceedings of the second annual ACM conference on Multimedia systems (MMSys) 2011*, pages 157–168, February 2011.

[6] L. Andrew, S. Hanly, S. Chan, and T. Cui. Adaptive deterministic packet marking. *IEEE Communications Letters*, 10(11):790–792, November 2006.

[7] G. Andrews. *The Theory of Partitions*. Cambridge Mathematical Library. Cambridge University Press, 1998.

[8] K. J. Astrom and R. M. Murray. *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, 2008.

[9] C. Aurrecoechea, A. T. Campbell, and L. Hauw. A survey of QoS architectures. *Multimedia Systems*, 6:138–151, May 1998.

[10] L. Badia, M. Levorato, and M. Zorzi. A channel representation method for the study of hybrid retransmission-based error control. *IEEE Transactions on Communications*, 57(7):1959–1971, July 2009.

[11] P. Balaouras and I. Stavrakakis. A self-adjusting rate adaptation scheme with good fairness and smoothness properties. *International Journal of Computer and Telecommunications Networking*, 48:829–855, August 2005.

[12] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable application layer multicast. *SIGCOMM Computer Communication Review*, 32(4):205–217, August 2002.

[13] M. Barcellos and P. Ezhilchelvan. An end-to-end reliable multicast protocol using polling for scaleability. In *Proceedings of the Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM) 1998*, volume 3, pages 1180–1187, March 1998.

[14] N. Bartolini, E. Casalicchio, and S. Tucci. A walk through content delivery networks. *Performance Tools and Applications to Networked Systems*, 2965:1–25, 2004.

[15] N. Benameur, S. B. Fredj, S. Oueslati-Boulahia, and J. W. Roberts. Quality of service and flow level admission control in the Internet. *International Journal of Computer and Telecommunications Networking*, 40:57–71, September 2002.

[16] R. Bernstein and S. Bernstein. *Schaum's Outline of Elements of Statistics II: Inferential Statistics*. Schaum's Outline Series. McGraw-Hill Companies,Incorporated, 1999.

[17] D. Bertsekas and R. Gallager. *Data networks*. Prentice-Hall, 1987.

[18] S. Biaz and N. H. Vaidya. Is the round-trip time correlated with the number of packets in flight? In *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement (IMC) 2003*, pages 273–278, October 2003.

[19] G. Bolch. *Queueing Networks And Markov Chains: Modeling and Performance Evaluation with Computer Science Applications*. Wiley-Interscience publication. 2006.

[20] J.-C. Bolot. End-to-end packet delay and loss behavior in the internet. *SIGCOMM Computer Communication Review*, 23:289–298, October 1993.

[21] J.-C. Bolot, S. Fosse-Parisis, and D. Towsley. Adaptive FEC-based error control for Internet telephony. In *Proceedings of the Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM) 1999*, volume 3, pages 1453–1460, March 1999.

[22] J.-C. Bolot, T. Turletti, and I. Wakeman. Scalable feedback control for multicast video distribution in the Internet. *SIGCOMM Computer Communication Review*, 24:58–67, October 1994.

[23] C. Bouras, A. Gkamas, and G. Kioumourtzis. Smooth multicast congestion control for adaptive multimedia transmission. In *Proceedings of the Conference on Next Generation Internet Networks (NGI) 2008*, pages 146–152, April 2008.

[24] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource ReSerVation Protocol (RSVP) – Version 1, Functional Specification. RFC 2205 (Proposed Standard), September 1997.

[25] L. Budzisz, R. Stanojevic, A. Schlote, F. Baker, and R. Shorten. On the fair coexistence of loss- and delay-based TCP. *IEEE/ACM Transactions on Networking*, 19(6):1811–1824, December 2011.

[26] S. Casner. Media Type Registration of Payload Formats in the RTP Profile for Audio and Video Conferences. RFC 4856 (Proposed Standard), February 2007.

[27] S. Cen, P. C. Cosman, and G. M. Voelker. End-to-end differentiation of congestion and wireless losses. *IEEE/ACM Transactions on Networking*, 11:703–717, October 2003.

[28] M. C. Chan and R. Ramjee. TCP/IP performance over 3G wireless links with rate and delay variation. In *Proceedings of the 8th annual international conference on Mobile computing and networking (ACM MobiCom) 2002*, pages 71–82, September 2002.

[29] B.-J. Chang, S.-Y. Lin, and J.-Y. Jin. Liad: Adaptive bandwidth prediction based logarithmic increase adaptive decrease for TCP congestion control in heterogeneous wireless networks. *International Journal of Computer and Telecommunications Networking*, 53:2566–2585, September 2009.

[30] M. Chen, J. Zhang, M. N. Murthi, and K. Premaratne. Delay-based TCP congestion avoidance: A network calculus interpretation and performance improvements. *International Journal of Computer and Telecommunications Networking*, 53:1319–1340, June 2009.

[31] M. Chiang, S. Low, A. Calderbank, and J. Doyle. Layering as optimization decomposition: A mathematical theory of network architectures. *Proceedings of the IEEE*, 95(1):255–312, January 2007.

[32] D. M. Chiu and A. S.-W. Tam. Fairness of traffic controls for inelastic flows in the Internet. *International Journal of Computer and Telecommunications Networking*, 51:2938–2957, August 2007.

[33] A. Chodorek and R. Chodorek. Streaming video over TFRC with linear throughput equation. *Advances in Electronics and Telecommunications*, 1:26–29, November 2010.

[34] J. Choi, J. Han, E. Cho, T. Kwon, and Y. Choi. A survey on content-oriented networking for efficient content delivery. *IEEE Communications Magazine*, 49(3):121–127, March 2011.

[35] M. Choi, M. Samokhina, K. Moklyuk, S. Choi, J. Heo, and S.-J. Oh. VPAL: Video packet adaptation layer for reliable video multicast over IEEE 802.11n WLAN. *International Journal for the Computer and Telecommunications Industry*, 33:2271–2281, December 2010.

[36] S. Choi and M. Handley. Designing TCP-friendly window-based congestion control for real-time multimedia applications. In *The 7th International Workshop on Protocols for Future, Large-Scale and Diverse Network Transports (PFLDNeT) 2009*, May 2009.

[37] C.-F. Chou, M.-W. Hsu, and C.-J. Lin. A trend-loss-density-based differential scheme in wired-cum-wireless networks. In *Proceedings of the International conference on Wireless Communications and Mobile Computing (IWCMC) 2006*, pages 239–244, July 2006.

[38] I. Cidon, A. Khamisy, and M. Sidi. Analysis of packet loss processes in high-speed networks. *IEEE Transactions on Information Theory*, 39(1):98–108, January 1993.

[39] D. Clark, L. Zhang, and S. Shenker. Supporting real-time applications in an integrated services packet network: Architecture and mechanism. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM) 1992*, 1992.

[40] D. D. Clark, M. L. Lambert, and L. Zhang. NETBLT: a high throughput transport protocol. *SIGCOMM Computer Communication Review*, 17(5):353–359, August 1987.

[41] D. Comer. *Internetworking with TCP/IP: Principles, protocols, and architecture.* Pearson Prentice Hall, 2006.

[42] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, editors. *Introduction to algorithms.* MIT Press, 2009.

[43] T. Cover and J. Thomas. *Elements of Information Theory.* Wiley Series in Telecommunications and Signal Processing. Wiley-Interscience, 2006.

[44] E. Dahlman. *3G evolution: HSPA and LTE for mobile broadband.* Electronics & Electrical. Elsevier Academic Press, 2007.

[45] P. H. Dana. Global positioning system (GPS) time dissemination for real-time applications. *Real-Time Systems - International Journal of Time-Critical Computing Systems*, 12(1):9–40, January 1997.

[46] L. De Cicco and S. Mascolo. TCP congestion control over 3g communication systems: An experimental evaluation of New Reno, BIC and Westwood+. *Lecture Notes in Computer Science*, 4712:73–85, 2007.

[47] L. De Cicco, S. Mascolo, and V. Palmisano. Feedback control for adaptive live video streaming. In *Proceedings of the second annual ACM conference on Multimedia systems (MMSys) 2011*, pages 145–156, February 2011.

[48] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460 (Draft Standard), December 1998.

[49] M. Dillinger, K. Madani, and N. Alonistioti. *Software Defined Radio: Architectures, Systems and Functions.* Wiley Series in Software Radio. Wiley, 2003.

[50] S. Eliason. *Maximum Likelihood Estimation: Logic and Practice.* SAGE Publications, 1993.

[51] E. O. Elliott. Estimates of error rates for codes on burst-noise channels. *Bell System Technical Journal*, 42:1977–1997, September 1963.

[52] ETSI. ETSI TS 302 304 v1.1.1 - Digital Video Broadcasting (DVB); transmission system for handheld terminals (DVB-H), 2004.

[53] ETSI. ETSI TS 102 542 v1.2.1 - Digital Video Broadcasting (DVB); guidelines for the implementation of DVB-IP Phase 1 specifications, 2008.

[54] D. Y. Eun and N. B. Shroff. Network decomposition: theory and practice. *IEEE/ACM Transactions on Networking*, 13:526–539, June 2005.

[55] E. Exposito, M. Gineste, L. Dairaine, and C. Chassot. Building self-optimized communication systems based on applicative cross-layer information. *Computer Standards and Interfaces*, 31:354–361, February 2009.

[56] J. Feng and L. Xu. On the time scale of TCP-friendly admission control protocols. In *Proceedings of the IEEE International Conference on Communications (ICC) 2008*, pages 45–51, May 2008.

[57] F. S. Filho, E. H. Watanabe, and E. de Souza e Silva. Adaptive forward error correction for interactive streaming over the Internet. In *Proceedings of the Global Communications Conference (GLOBECOM) 2006*, 2006.

[58] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-based congestion control for unicast applications. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM) 2000*, pages 43–56, September 2000.

[59] S. Floyd and T. Henderson. The NewReno modification to TCP's fast recovery algorithm. RFC 2582 (Experimental), April 1999.

[60] S. Floyd, V. Jacobson, C.-G. Liu, S. McCanne, and L. Zhang. A reliable multicast framework for light-weight sessions and application level framing. *IEEE/ACM Transactions on Networking*, 5:784–803, December 1997.

[61] T. Friedman and D. F. Towsley. Multicast session membership size estimation. In *Proceedings of the Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM) 1999*, pages 965–972, March 1999.

[62] M. Garetto and D. Towsley. Modeling, simulation and measurements of queuing delay under long-tail internet traffic. In *Proceedings of the ACM international conference on measurement and modeling of computer systems (SIGMETRICS) 2003*, pages 47–57, 2003.

[63] L. L. Garlick, R. Rom, and J. B. Postel. Reliable host-to-host protocols: Problems and techniques. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM) 1977*, pages 4.58–4.65, 1977.

[64] L. Gharai, C. Perkins, and T. Lehman. Packet reordering, high speed networks and transport protocol performance. In *Proceedings of the 13th International Conference on Computer Communications and Networks (ICCCN) 2004*, pages 73–78, October 2004.

[65] E. N. Gilbert. Capacity of a burst-noise channel. *Bell System Technical Journal*, 39:1253–1265, September 1960.

[66] A. Goldsmith. *Wireless Communications*. Cambridge University Press, 2005.

[67] M. Gorius, Y. Shuai, and T. Herfet. Dynamic media streaming under predictable reliability. In *Proceedings of the IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB) 2012*, June 2012.

[68] S. Ha, I. Rhee, and L. Xu. CUBIC: a new TCP-friendly high-speed TCP variant. *ACM SIGOPS operating systems review*, 42:64–74, July 2008.

[69] F. Hartanto and H. Sirisena. Hybrid error control mechanism for video transmission in the wireless IP networks. In *Proceedings of the 10th IEEE Workshop on Local and Metropolitan Area Networks 1999*, pages 126–132, November 1999.

[70] D. A. Hayes and G. Armitage. Revisiting TCP congestion control using delay gradients. In *Proceedings of the 10th international IFIP TC 6 conference on Networking (NETWORKING) 2011*, pages 328–341, May 2011.

[71] J. Heidemann, K. Obraczka, and J. Touch. Modeling the performance of HTTP over several transport protocols. *IEEE/ACM Transactions on Networking*, 5(5):616–630, October 1997.

[72] T. Herfet. Future media Internet: Video and audio transport - a new paradigm, October 2011.

[73] S. Heubach and T. Mansour. *Combinatorics of Compositions and Words*. Discrete Mathematics and its Applications. CRC Press, 2009.

[74] I. Hofmann, N. Farber, and H. Fuchs. A study of network performance with application to adaptive HTTP streaming. In *Proceedings of the IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB) 2011*, pages 1–6, June 2011.

[75] H. W. Holbrook, S. K. Singhal, and D. R. Cheriton. Log-based receiver-reliable multicast for distributed interactive simulation. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM) 1995*, pages 328–341, August 1995.

[76] D. Hong, C. Albuquerque, C. Oliveira, and T. Suda. Evaluating the impact of emerging streaming media applications on TCP/IP performance. *IEEE Communications Magazine*, 39(4):76–82, April 2001.

[77] M. Hosseini, D. Ahmed, S. Shirmohammadi, and N. Georganas. A survey of application-layer multicast protocols. *IEEE Communications Surveys Tutorials*, 9(3):58–74, April 2007.

[78] ISO/IEC. ISO/IEC 13818-1:2007 - generic coding of moving pictures and associated audio information: Systems, 2007.

[79] V. Jacobson. Congestion avoidance and control. *SIGCOMM Computer Communication Review*, 18:314–329, August 1988.

[80] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard. Networking named content. In *Proceedings of the 5th international Conference on emerging Networking Experiments and Technologies (CoNExT) 2009*, pages 1–12, 2009.

[81] R. Jain. A delay-based approach for congestion avoidance in interconnected heterogeneous computer networks. *SIGCOMM Computer Communications Review*, 19:56–71, October 1989.

[82] R. Jain, D. Chiu, and W. Hawe. *A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer System.* Eastern Research Laboratory, Digital Equipment Corporation, 1984.

[83] H. Jung, S. G. Kim, H. Y. Yeom, S. Kang, and L. Libman. Adaptive delay-based congestion control for high bandwidth-delay product networks. In *Proceedings of the 31th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM) 2011*, pages 2885–2893, April 2011.

[84] S. A. Karim and P. Hovell. Everything over IP - an overview of the strategic change in voice and data networks. *BT Technology Journal*, 17(2):24–30, April 1999.

[85] R. G. Kermode. Scoped hybrid automatic repeat request with forward error correction (SHARQFEC). In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM) 1998*, August 1998.

[86] J. Kingman. *Poisson Processes.* Oxford University Press, 1993.

[87] E. Kohler, M. Handley, and S. Floyd. Designing DCCP: congestion control without reliability. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM) 2006*, pages 27–38, August 2006.

[88] J. Kurose and K. Ross. *Computer Networking: A Top-Down Approach.* Pearson, 2010.

[89] A. Kuzmanovic. The power of explicit congestion notification. *SIGCOMM Computer Communication Review*, 35:61–72, August 2005.

[90] Y.-C. Lai and C.-N. Lai. DCCP partial reliability extension with sequence number compensation. *International Journal of Computer and Telecommunications Networking*, 52:3085–3100, November 2008.

[91] D. Leith, R. Shorten, G. McCullagh, L. Dunn, and F. Baker. Making available base-RTT for use in congestion control applications. *IEEE Communications Letters*, 12(6):429–431, June 2008.

[92] A. Li. RTP Payload Format for Generic Forward Error Correction. RFC 5109 (Proposed Standard), Dec. 2007.

[93] S.-Y. Li, R. Yeung, and N. Cai. Linear network coding. *IEEE Transactions on Information Theory*, 49(2):371–381, February 2003.

[94] Z. Li. *Multicast MAC extensions for high rate real-time traffic in wireless LANs.* PhD thesis, Saarland University, 2011.

[95] Z. Li and T. Herfet. MAC layer multicast error control for IPTV in wireless LANs. *IEEE Transactions on Broadcasting*, 55(2):353–362, June 2009.

[96] R. Lidl and H. Niederreiter. *Finite Fields.* Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1996.

[97] S. Liu, T. Başar, and R. Srikant. TCP-Illinois: a loss and delay-based congestion control algorithm for high-speed networks. In *Proceedings of the 1st international conference on performance evaluation methodolgies and tools (valuetools) 2006*, October 2006.

[98] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim. A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications Surveys Tutorials*, 7(2):72–93, 2005.

[99] D. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

[100] D. MacKay. Fountain codes. *IEEE Proceedings in Communications*, 152(6):1062–1068, December 2005.

[101] G. Marfia and M. Roccetti. TCP at last: reconsidering TCP's role for wireless entertainment centers at home. *IEEE Transactions on Consumer Electronics*, 56(4):2233–2240, November 2010.

[102] D. Marinescu and G. Marinescu. *Classical and Quantum Information*. Academic Press, 2011.

[103] G. McCullagh and D. Leith. Delay-based congestion control: Sampling and correlation issues revisited. Technical report, 2008.

[104] S. Meyn and R. L. Tweedie. *Markov Chains and Stochastic Stability*. Cambridge University Press, 2009.

[105] F. P. Miller, A. F. Vandome, and J. McBrewster. *Cyclic Redundancy Check: Computation of CRC*. Alpha Press, 2009.

[106] D. Mills, J. Martin, J. Burbank, and W. Kasch. Network Time Protocol Version 4: Protocol and Algorithms Specification. RFC 5905 (Proposed Standard), June 2010.

[107] B. Mukherjee and T. Brecht. Time-lined TCP for the TCP-friendly delivery of streaming media. In *Proceedings of the International Conference on Network Protocols (ICNP) 2000*, November 2000.

[108] S. Muraoka, H. Masuyama, S. Kasahara, and Y. Takahashi. Performance analysis of FEC recovery using finite-buffer queueing system with general renewal and poisson inputs. In *Proceedings of the 20th international teletraffic conference on managing traffic performance in converged networks (ITC) 2007*, pages 707–718, June 2007.

[109] K. Nichols, S. Blake, F. Baker, and D. Black. Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers. RFC 2474 (Proposed Standard), December 1998.

[110] J. Nonnenmacher and E. Biersack. Scalable feedback for large groups. *IEEE/ACM Transactions on Networking*, 7(3):375–386, June 1999.

[111] J. Nonnenmacher and E. W. Biersack. Reliable multicast: Where to use FEC. In *Proceedings of the 5th international workshop on Protocols for High-Speed Networks (PfHSN) 1996*, October 1996.

[112] B. Oh, J. Han, and J. Lee. Timer and sequence based packet loss detection scheme for efficient selective retransmission in DCCP. *Communication and Networking*, 119:112–120, 2010.

[113] J. Opdyke. A unified approach to algorithms generating unrestricted and restricted integer compositions and integer partitions. *Journal of Mathematical Modelling and Algorithms*, 9:53–97, 2010.

[114] J. Ott, S. Wenger, N. Sato, C. Burmeister, and J. Rey. Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF). RFC 4585 (Proposed Standard), July 2006.

[115] S. Pack, X. Shen, J. Mark, and L. Cai. A two-phase loss differentiation algorithm for improving TFRC performance in IEEE 802.11 WLANs. *IEEE Transactions on Wireless Communications*, 6(11):4164–4175, November 2007.

[116] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP throughput: A simple model and its empirical validation. Technical report, 1998.

[117] C. Partridge. Implementing the reliable data protocol (RDP). In *Proceedings of the USENIX Summer Conference*, pages 367–380, June 1987.

[118] S. Paul, K. K. Sabnani, J. C. Lin, and S. Bhattacharyya. Reliable multicast transport protocol (RMTP). *IEEE Journal on Selected Areas in Communications*, 15(3):407–421, April 1997.

[119] V. Paxson. End-to-end internet packet dynamics. *IEEE/ACM Transactions on Networking*, 7(3):277–292, June 1999.

[120] L. Peterson and B. Davie. *Computer Networks: A Systems Approach*. The Morgan Kaufmann Series in Networking. Elsevier Science, 2007.

[121] S. Pingali, D. Towsley, and J. F. Kurose. A comparison of sender-initiated and receiver-initiated reliable multicast protocols. *SIGMETRICS Performance Evaluation Review*, 22:221–230, May 1994.

[122] T. Porter and X.-H. Peng. Effective video content distribution by combining TCP with adaptive FEC coding. In *Proceedings of the IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB) 2010*, pages 1–5, March 2010.

[123] J. Postel. User Datagram Protocol. RFC 768 (Standard), August 1980.

[124] J. Postel. Internet Protocol. RFC 791 (Standard), September 1981.

[125] J. Postel. Transmission Control Protocol. RFC 793 (Standard), September 1981.

[126] J. Postel and C. Anderson. White Pages Meeting Report. RFC 1588 (Informational), February 1994.

[127] R. Prasad and M. Ruggieri. *Technology trends in wireless communications*. Artech House universal personal communications series. Artech House, 2003.

[128] M. Rajiullah and A. Brunstrom. On the effectiveness of PR-SCTP in networks with competing traffic. In *Proceedings of the IEEE Symposium on Computers and Communications (ISCC) 2011*, pages 898–905, July 2011.

[129] L. Rizzo. Effective erasure codes for reliable computer communication protocols. *SIGCOMM Computer Communication Review*, 27:24–36, April 1997.

[130] L. Roychoudhuri and E. S. Al-Shaer. Real-time packet loss prediction based on end-to-end delay variation. *IEEE Transactions on Network and Service Management*, 2(1):29–38, November 2005.

[131] D. Rubenstein, S. Kasera, D. Towsley, and J. Kurose. Improving reliable multicast using active parity encoding services (APES). In *Proceedings of the Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM) 1999*, volume 3, pages 1248–1255, March 1999.

[132] D. Rubenstein, J. Kurose, and D. Towsley. A study of proactive hybrid FEC/ARQ and scalable feedback techniques for reliable, real-time multicast. *International Journal for the Computer and Telecommunications Industry*, 24:563–574, 2001.

[133] W. Ryan and S. Lin. *Channel codes: classical and modern*. Cambridge University Press, 2009.

[134] M. v. d. Schaar and P. A. Chou. *Multimedia over IP and Wireless Networks: Compression, Networking, and Systems*. Academic Press, Inc., 2007.

[135] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. RFC 3550 (Standard), July 2003.

[136] H. Seferoglu, A. Markopoulou, U. Kozat, M. Civanlar, and J. Kempf. Dynamic FEC algorithms for TFRC flows. *IEEE Transactions on Multimedia*, 12(8):869–885, December 2010.

[137] N. Seitz. ITU-T QoS standards for IP-based networks. *IEEE Communications Magazine*, 41(6):82–89, June 2003.

[138] E. Setton and B. Girod. Congestion-distortion optimized scheduling of video over a bottleneck link. In *Proceedings of the 6th IEEE Workshop on Multimedia Signal Processing 2004*, pages 179–182, October 2004.

[139] C. E. Shannon. A mathematical theory of communication. *SIGMOBILE Mobile Computing Communications Review*, 5(1):3–55, January 2001.

[140] S. Shenker. Fundamental design issues for the Future Internet. *IEEE Journal on Selected Areas in Communications*, pages 1176–1188, 1995.

[141] H.-P. Shiang and M. van der Schaar. Content-aware TCP-friendly congestion control for multimedia transmission. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) 2011*, pages 2356–2359, May 2011.

[142] A. Shokrollahi. Raptor codes. *IEEE/ACM Transactions on Networking*, 14:2551–2567, June 2006.

[143] R. Stewart, I. Arias-Rodriguez, K. Poon, A. Caro, and M. Tuexen. Stream Control Transmission Protocol (SCTP) Specification Errata and Issues. RFC 4460 (Informational), April 2006.

[144] T. Stockhammer. Dynamic adaptive streaming over HTTP – standards and design principles. In *Proceedings of the second annual ACM conference on Multimedia systems (MMSys) 2011*, pages 133–144, February 2011.

[145] H. Sze, S. Liew, and Y. Lee. A packet-loss-recovery scheme for continuous-media streaming over the Internet. *IEEE Communications Letters*, 5(3):116–118, March 2001.

[146] G. Tan. *Optimum hybrid error correction scheme under strict delay constraints*. PhD thesis, Saarland University, 2009.

[147] G. Tan and T. Herfet. A novel adaptive hybrid error correction scheme for wireless DVB services. *International Journal of Computer Science and Network Security*, 1(2):187–198, 2008.

[148] K. Tan and J. Song. Compound TCP: A scalable and TCP-friendly congestion control for high-speed networks. In *Proceedings of the 4th International workshop on Protocols for Fast Long-Distance Networks (PFLDNet) 2006*, May 2006.

[149] S. Tao and R. Guerin. On-line estimation of internet path performance: an application perspective. In *Proceedings of the Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM) 2004*, volume 3, pages 1774–1785, March 2004.

[150] O. Tickoo, V. Subramanian, S. Kalyanaraman, and K. K. Ramakrishnan. LT-TCP: End-to-end framework to improve TCP performance over networks with lossy channels. In *Proceedings of the 13th International Workshop on Quality of Service (IWQoS) 2005*, pages 81–93, June 2005.

[151] D. T. Tran and E. Choi. A reliable UDP for ubiquitous communication environments. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM) 2007*, pages 1–6, August 2007.

[152] M. Trott and M. Trott. *The Mathematica guidebook for programming*. The Mathematica Guidebook for Symbolics. Telos-Springer, 2004.

[153] D. S. Turaga and T. Chen. Hierarchical modeling of variable bit rate video sources. In *Proceedings of the 11th International Workshop on Packet Video (PV) 2001*, page 2001, April 2001.

[154] M. van der Schaar, S. Krishnamachari, S. Choi, and X. Xu. Adaptive cross-layer protection strategies for robust scalable video transmission over 802.11 WLANs. *IEEE Journal on Selected Areas in Communications*, 21(10):1752–1763, December 2003.

[155] O. Verscheure, P. Frossard, and M. Hamdi. MPEG-2 video services over packet networks: Joint effect of encoding rate and data loss on user-oriented QoS. In *Proceedings of the 8th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV) 1998*, pages 257–264, 1998.

[156] J. Vicente and D. Hutchison. *Management of Multimedia Networks and Services.* Lecture Notes in Computer Science. Springer, 2004.

[157] K. Vlachos. Burstification effect on the TCP synchronization and congestion window mechanism. In *Fourth International Conference on Broadband Communications, Networks and Systems, (BROADNETS) 2007*, pages 24–28, September 2007.

[158] B. Wang, J. Kurose, P. Shenoy, and D. Towsley. Multimedia streaming via TCP: an analytic performance study. In *Proceedings of the 12th annual ACM international conference on Multimedia (MULTIMEDIA) 2004*, pages 908–915, October 2004.

[159] A. Weaver. Xpress transport protocol version 4. In *Proceedings of the IEEE International Workshop on Factory Communication Systems (WFCS) 1995*, pages 165–174, October 1995.

[160] D. X. Wei, C. Jin, S. H. Low, and S. Hegde. Fast TCP: motivation, architecture, algorithms, performance. *IEEE/ACM Transactions on Networking*, 14:1246–1259, December 2006.

[161] D. Wessels and K. Claffy. Application of Internet cache protocol (ICP), version 2. RFC 2187 (Informational), September 1997.

[162] J. Widmer. *Equation-Based Congestion Control for Unicast and Multicast Data Streams.* PhD thesis, University of Mannheim, May 2003.

[163] J. Widmer and M. Handley. Extending equation-based congestion control to multicast applications. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM) 2001*, pages 275–285, August 2001.

[164] S. Wu, S. Banerjee, and X. Hou. Performance evaluation and comparison of multicast feedback control mechanisms. *Simulation*, 82(5):345–362, May 2006.

[165] X. Xu, A. Myers, H. Zhang, and R. Yavatkar. Resilient multicast support for continuous-media applications. In *Proceedings of the IEEE 7th International Workshop on Network and Operating System Support for Digital Audio and Video 1997*, pages 183–194, May 1997.

[166] M. Yajnik, S. Moon, J. Kurose, and D. Towsley. Measurement and modelling of the temporal dependence in packet loss. In *Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM) 1999*, volume 1, pages 345–352, March 1999.

[167] J. Yee and J. Weldon, E.J. Evaluation of the performance of error-correcting codes on a Gilbert channel. *IEEE Transactions on Communications*, 43(8):2316–2323, August 1995.

[168] Y. Yi and M. Chiang. Stochastic network utility maximisation - a tribute to kellys paper published in this journal a decade ago. *European Transactions on Telecommunications*, 19(4):421–442, 2008.

[169] X. Yu, J. Modestino, and X. Tian. The accuracy of Gilbert models in predicting packet-loss statistics for a single-multiplexer network model. In *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM) 2005*, volume 4, pages 2602–2612, March 2005.

[170] X. Yu, J. W. Modestino, R. Kurceren, and Y. S. Chan. A model-based approach to evaluation of the efficacy of FEC coding in combating network packet losses. *IEEE/ACM Transactions on Networking*, 16:628–641, June 2008.

[171] M. Yuksel, K. K. Ramakrishnan, R. Doverspike, R. Sinha, G. Li, K. Oikonomou, and D. Wang. Cross-layer techniques for failure restoration of IP multicast with applications to IPTV. In *Proceedings of the 2nd international conference on Communication Systems and Networks (COMSNETS) 2010*, pages 223–232, January 2010.

[172] Q. Zhang, W. Zhu, and Y.-Q. Zhang. Resource allocation for multimedia streaming over the internet. *IEEE Transactions on Multimedia*, 3(3):339–355, September 2001.