Optimized Wide Area Media Transport Strategies

Dissertation

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

vorgelegt von

Michael Karl

Saarbrücken, April 2015

Tag des Kolloquiums

30. September 2015

Dekan der Naturwissenschaftlich-Technischen Fakultät I

Prof. Dr. Markus Bläser
Universität des Saarlandes, Saarbrücken, Deutschland

Prüfungsausschuss

Prof. Dr. Dietrich Klakow (Vorsitzender des Prüfungsausschusses)
Universität des Saarlandes

Prof. Dr. Thorsten Herfet (Berichterstatter)
Universität des Saarlandes

Prof. Dr. Jens Krüger (Berichterstatter)
Universität Duisburg-Essen

Dr.-Ing. Sven Gehring (Akademischer Beisitzer)
Deutsches Forschungsinstitut für künstliche Intelligenz GmbH

Short Abstract

Modern networks are increasingly used for a multitude of different application scenarios, which primarily include multimedia transmission with a strict delivery time constraint. This kind of content has far more complex requirements than traditional applications, such as file transfer via download or sending an email. This thesis motivates the segmentation of default IP-based communication paths for optimized network and transport utilization concerning retransmission error coding schemes and strict delivery time limits. Theoretical considerations for general channel coding approaches with no time limitation show, that the network load measurably decreases in case the transmission is fully split. Time-restricted transmissions with packet-retransmission error correction schemes require more complex mechanisms for the time budget distribution and retransmission characteristics. The development of an upper bound for the number of segments established on a transmission path with hybrid error correction and a time constraint represents a significant contribution of this thesis. Metrics are presented that enable a reliable identification of network segments with a high optimization potential. Mathematical programming and graph theoretical methods demonstrate their versatility. Experimental measurements and simulations verify that network path segmentation leads to a significant reduction of the network load.

Kurze Zusammenfassung

Moderne Netzwerke werden zunehmend für multimediale Übertragungen mit strikter Zeitbegrenzung verwendet. Diese haben weitaus komplexere Anforderungen als traditionelle Anwendungen wie die Übertragungen von Dateien oder der Versand von Emails. Die vorliegende Arbeit motiviert die Segmentierung traditioneller IP-basierter Kommunikationspfade zur Optimierung des Netzwerktransports unter Verwendung von paketbasierten Fehlerschutzmechanismen und fixen Begrenzungen der zulässigen Latenz. Theoretische Betrachtungen für Kanalkodierungen ohne Zeitbegrenzung zeigen eine erkennbare Verbesserung der Netzwerkauslastung, sofern Übertragungspfade vollständig segmentiert werden. Zeitkritische Übertragungen mit paketbasierten Korrekturverfahren erfordern komplexe Mechanismen für Zeitbudgetverteilung und Paketwiederholungsverhalten. Die Ausarbeitung einer oberen Schranke für die Anzahl von Segmenten innerhalb eines Übertragungspfades bei Verwendung von hybriden Fehlerschutzverfahren unter Zeitbeschränkung stellt einen signifikanten Beitrag der Arbeit dar. Des Weiteren werden Metriken vorgestellt, die eine verlässliche Lokalisierung von Netzwerksegmenten mit hohem Optimierungspotential erlauben. Methoden der mathematischen Programmierung und Graphentheorie demonstrieren die praktische Einsetzbarkeit. Experimentelle Auswertungen und Simulationen belegen, dass die Segmentierung eines Transportpfades zu einer signifikanten Reduktion der Netzwerkauslastung führt.

Abstract

The world is connected by global wide area networks that help to exchange a multitude of different content types. Most of them are packet-switched and use the popular *Internet Protocol* to exchange information. In the last decade this protocol has established itself as a flexible and widely deployed abstraction layer for communication. Consequently, the trend is towards an all IP infrastructure, where the majority of communication environments use the IP layer. This eases the deployment of globally distributed applications, as the access between different systems is significantly harmonized.
Besides the infrastructure data content types have also changed. An interesting and challenging content type is multimedia. New technologies and inventions have caused a significant increase in the number of multimedia application, and according to forecasts it will represent the majority of data in global networks by the end of 2020. Multimedia content introduces strict requirements to its transport as human receivers link specific expectations to this application content.

In this thesis ARQ-based network optimization approaches are presented that focus on the exploitation of transport infrastructures in terms of time constraint multimedia content. Packet-switched networks have generally inhomogeneous topologies that provide a reasonable basis for a spatial transport optimization scheme called *Loss Domain Separation (LDS)*. With LDS network paths are divided into multiple segments according to their characteristics, and finally individual correction schemes are applied to those segments. As these error correction mechanisms can be efficiently tailored to the segments, this leads to higher transport efficiencies as transmission latency, congestion, and required transmission data rates are reduced. Such a segmentation breaks with the traditional end-to-end transport and correction schemes. It is shown that in case of an unrestricted delivery time a full segmentation, i.e. as many as possible segments, represents the optimum environmental setup. However, in case of a time restricted transmission scenario, there exists a *Saturation Point* which corresponds to the upper limit of segments on a specific path. Additionally, a set of mapping approaches is presented. It focuses on an evolutionary application of multi-dimensional metric spaces in default network architectures, as an efficient multimedia transmission requires the consideration of multiple characteristics rather than only a single. Legacy network environments are not aware of the transported content type, and the coordination between network infrastructure and source application is not satisfying. Hence, a network collaboration scheme is presented that enables a fine granular information exchange between both parties in order to achieve flexible and reliable multimedia transports. In particular a set of *Linear Programming* models is frequently used, as these models offer a suitable optimization basis and adequately describe network and application requirements. Ultimately two demonstrators are presented that have been developed during this thesis and constitute a flexible basis for *Loss Domain Separation* research.

Contents

List of Figures

List of Tables

Nomenclature

3GPP	3rd Generation Partnership Project
ACK	Acknowledgment
ACL	Access Control List
AHEC	Adaptive Hybrid Error Correction
AL-FEC	Application Layer Forward Error Correction
ALF	Application Level Framing
API	Application Programming Interface
APSP	All Pairs Shortest Path
ARP	Address Resolution Protocol
ARQ	Automatic Repeat Request
AS	Autonomous System
ATM	Asynchronous Transfer Mode
BFS	Breadth-First Search
BGP	Border Gateway Protocol
BIP	Binary Integer Programming
CDN	Content Delivery Network
CPU	Central Processing Unit
CSI	Channel State Information
DASH	Dynamic Adaptive Streaming over HTTP
DHCP	Dynamic Host Configuration Protocol
DiffServ	Differentiated Services
DLCI	Direct Link Connection Identifier
DNS	Domain Name System
DPI	Deep Packet Inspection
DSCP	Differentiated Services Code Point
DVB	Digital Video Broadcasting
DVB-IPTV	Digital Video Broadcasting - IPTV
DVMRP	Distance Vector Multicast Routing Protocol

E2E	End-to-End
ECR	Error Correction Relay
ETSI	European Telecommunications Standards Institute
FEC	Forward Error Correction
FR	Frame Relay
FTP	File Transfer Protocol
FTTx	Fiber to the x
GPU	Graphics Processing Unit
HARQ	Hybrid Automatic Repeat Request
HbbTV	Hybrid Broadcast Broadband Television
HEC	Hybrid Error Correction
HTML	Hypertext Markup Language
HTTP	Hypertext Transport Protocol
ICMP	Internet Control Message Protocol
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IGMPv3	Internet Group Management Protocol (Version 3)
ILP	Integer Linear Programming
IMS	IP Multimedia Subsystem
IntServ	Integrated Services
IP	Internet Protocol / Integer Programming
IPDV	IP Delay Variation
IPER	IP Error Ratio
IPI	IP Infrastructure
IPLR	IP Loss Ratio
IPRR	IP Reordering Ratio
IPTD	IP Transfer Delay
IPTV	Internet Protocol Television
IPv4	IP Version 4
IPv6	IP Version 6
IS-IS	Intermediate System to Intermediate System
ISO/OSI	International Standardization Organization
ISP	Internet Service Provider
ITU	International Telecommunications Union
IWF	Interworking Function
IWU	Interworking Units
KP	Knapsack Problem
LAG	Link Aggregation Group

LAN	Local Area Network
LCN	Logical Channel Number
LDS	Loss Domain Separation
LLDP	Link-layer Discovery Protocol
LP	Linear Programming
LWAN	Local Wide Area Network (Demonstrator)
MAC	Media Access Control
MCKP	Multiple-Choice Knapsack Problem
MILP	Mixed Integer Linear Programming
MLE	Maximum-Likelihood Estimation
MOSPF	Multicast Open Shortest Path First
MPEG	Motion Picture Experts Group
MPLS	Multi-Protocol Label Switching
MVC	Model View Controller
NACK	Negative Acknowledgment
NF	Network Function
NFV	Network Function Virtualization
NFV-FG	Network Function Virtualization Forwarding Graph
NFVI	Network Function Virtualization Infrastructure
NGN	Next Generation Networks
NIC	Network Interface Card
NIMFO	Network Information Managed Flow Optimization
NP	Nondeterministic Polynomial time
NRCS	Network Rate Control Service
NSCA	Network Supported Congestion Avoidance
OOB	Out of Band
OSIE	Open System Interconnection Environment
OSPF	Open Shortest Path First
OXM	OpenFlow Extensible Match
P2P	Peer-to-Peer
PBX	Private Branch Exchange
PEP	Performance Enhancing Proxy
PHB	Per Hop Behavior
PIM	Protocol Independent Multicast
PLR	Packet Loss Rate
POTS	Plain Old Telephone System
PRRT	Predictably Reliable Real-Time Transport
PSTN	Public Switch Telephone Network
QoE	Quality of Experience

QoS	Quality of Service
RED	Random Early Drop
RFC	Request for Comments
RI	Redundancy Information
RMA	Reliability, Maintenance, Accessibility
RPL	Routing Protocols in Low-Power and Lossy Networks
RSVP	Resource Reservation Protocol
RSVP-TE	Resource Reservation Protocol (Traffic Engineering Extension)
RTCP	Real-Time Control Protocol
RTP	Real-Time Transport Protocol
RTP/AVP	RTP Audio Video Profile
RTSP	Real-Time Setup Protocol
RTT	Round Trip Time
SCIP	Solving Constraint Integer Program
SDG	Supplementary Data Gateway
SDN	Software Defined Networking
SIP	Session Initiation Protocol
SMPTE	Society of Motion Picture and Television Engineers
SNR	Signal to Noise Ratio
SPI	Stateful Packet Inspection
SSSP	Single-Source Shortest Path
STB	Set Top Box
TAMCRA	Tunable Accuracy Multiple Constraints Routing Algorithm
TCP	Transport Control Protocol
TE	Traffic Engineering
TFRC	TCP Friendly Rate Control
TISPAN	Telecommunications and Internet converged Services and Protocols for Advanced Networking
TLV	Type Length Value
TOS	Type of Service
TTL	Time to Live
TV	Television
UDP	User Datagram Protocol
VCI	Virtual Channel Identifier
VLAN	Virtual Local Area Network
VNF	Virtualized Network Function
VNI	Virtual Networking Index
VoATM	Voice over ATM
VoD	Video on Demand

VoFR	Voice over Frame Relay
VoIP	Voice over Internet Protocol
VPI	Virtual Path Identifier
WAN	Wide Area Network
WFQ	Weighted Fair Queuing
xDSL	x Digital Subscriber Line
ZIMPL	Zuse Institute Mathematical Language

Chapter 1

Introduction

The domain of telecommunication has become a major foundation of today's society. It has a significant impact on nearly all aspects of everyday's life as a remarkable number of devices is already connected to the Internet, that use a large set of different connection channels. It is foreseeable, that in the near future the majority of communication channels are based on the Internet Protocol [1] and packet-switched network infrastructures. They represent a flexible and reliable abstraction layer to transport a large number of different content types. With the IP infrastructure, it becomes practical to converge the totality of telecommunication structures to a single scheme. Today's most prominent global IP infrastructure is the Internet, that allows an around the clock access at nearly every geographical location to distributed information.

Besides the essential opportunities provided by data networks, the content that is exchanged between different communication parties has also substantially changed. In the beginnings of the Internet, the major part of data was text-based, such as email, FTP, or simple and static HTML web pages. This type of content has modest requirements to its transport conditions. In the course of time, the information being exchanged has continuously moved to the multimedia domain. In 2015 it is common to use the Internet to make phone calls or to retrieve any video material from a hardly estimable number of available servers. Thereby, it does not make a difference if the data is retrieved via static wired infrastructures or via mobile wireless channels. Major representatives for multimedia application systems are commercial Internet-TV (e.g. IPTV), Video on Demand (VoD), WebTV, Internet telephony, or broadcasting services. Multimedia content introduces new requirements to the transmission. This includes a reliable delivery that does not exceed an individual upper limit for the residual error rate and the delivery latency. If multimedia content is not delivered considering its special requirements, the

receiving user does not achieve the desired quality and the transmission is seen as impractical. This shows how the end user's demands have remarkably changed with the introduction of new application types. These expectations constitute a fundamental contrast to the legacy application requirements for websites or emails. To emphasize the ongoing paradigm shift in Internet communications, Cisco's Visual Networking Index (VNI) forecasts that approximately 80% of the global consumer traffic will be IP video content in 2018[1].

In addition to the content, also the network infrastructures have been modified, extended, and experienced. The decentrally organized packet-switched IP networks still represent the default basis for the global data exchange. Thereby, variable-sized chunks of data are individually forwarded from the source to the sink without respecting their individual requirements. This increases the tolerance for single point of failure situations and consequently the reliability and stability of the global infrastructures. This legacy architecture is considered to be inflexible regarding multimedia transmissions. Individual data chunks may take different routes through the network, thus facing distinct delays, packet loss probabilities, and other transmission-specific influences. Network protocols for multimedia data and related error correction schemes address these environmental characteristics, and adequately customize the content transmission to the present network parameters.

Traditional network protocols are designed to either provide a fast but unreliable, or a perfectly reliable transmission, while introducing a potentially unlimited delay. For high quality multimedia content the traditional network protocols and their mechanisms are not sufficient. They have been developed and introduced for low transmission data rates and nearly unbounded latencies, as it is acceptable for web services and emails. These protocols map complex network infrastructure between a sender and a source into an abstract virtual segment, i.e. both communication partners are seen to be directly connected via a single link. This represents a fundamental simplification of the present infrastructure and avoids highly granular transmission approaches, that lead to an efficient data exchange. The transport of multimedia content via these legacy transport protocols is possible, but introduces a set of side-effects that must be managed. As an example, if Internet telephony is realized by unreliable UDP [2], it requires a complex application error correction scheme to handle appearing transmission errors. For TCP [3], it shows that the application can profit from the integrated error correction mechanism, but this introduces larger delivery delays. Consequently, it must implement larger buffers and cope with higher end-to-end transmission latencies.

[1] http://www.cisco.com/c/en/us/solutions/collateral/service-provider/ip-ngn-ip-next-generation-network/white_paper_c11-481360.html (2015/01/29)

The efficient and optimized utilization of network infrastructures is an inherent requirement for the future media Internet. This includes that all participating components, such as the network infrastructures, the transport protocols, and the applications, are harmonized and work closely together to obtain an optimized multimedia transport.

1.1 Multimedia Transport

The optimization of network environments for the transport of multimedia content is a significant task for modern transport infrastructures. Networks must ensure a reliable, timely, and continuous delivery, while efficiently exploiting the available network resources.

In current environments a highly inhomogeneous set of different applications and network technologies is available. This includes the concurrent data streaming of Internet telephony and video content, a multitude of text-based services, file downloads, and a variety of network services, such as DNS [4, 5]. All these applications are contacted via different access technologies, e.g. mobile radio or cable connections. This immense correlation of applications and infrastructure leads to an inevitable demand for optimized and individual transmission approaches, that yield an efficient network resource utilization. Most applications demand for a certain level of quality, e.g. a maximum transmission latency, or an upper limit for the residual error probability.

Early networks have been *Public Switched Telephone Networks (PSTN)* [6], based on a circuit-switched architecture. PSTNs have the prominent characteristic, that the number of end devices is large compared to the available network capacity. This consequently leads to the need for an adequate admission control scheme, as calls are only established in case sufficient end-to-end capacity is available at calling time. Although PSTNs occupy a complete network path between the communication endpoints and simultaneously reserve the required capacity for the transmission, they are inferior to modern packet-based infrastructures.

Today's networks are constantly converging towards an infrastructure that is based on the flexible and prevalent IP [1] specification. Applications facing packet-switched transport architectures, and as IP packets can be delivered to nearly every end-device, it is seen as the future's global communication infrastructure. The most prominent representative of IP networks is the *Internet*. Packet-switched networks significantly differ from circuit-switched networks. Accordingly, technologies, strategies, and principles for transportation of content must be reconsidered. Multimedia data with its special

requirements is a valid example for this demand. Packet-switched environments are traditionally based on the best-effort principle, i.e. packets are individually forwarded in the network without relation to any other component. This principle is seen as suboptimal for the transport of multimedia content, although it excellently works with legacy applications and moderate requirements. Multimedia requires special and individual mechanisms to be transmitted over packet-switched networks, in order to obtain an acceptable delivery procedure. Consequently, a transport optimization scheme is required for multimedia content that incorporates all aspects of a communication.

Optimization implies a mechanism that finds acceptable parameters for a system with multiple linked components. Thereby, it uses available knowledge and a set of side information.

In the domain of network traffic, this leads to an optimization of a system including transport protocols, network infrastructure, and the applications, that represent source and sink. Each of these components individually participates in a transmission of data, provides a special functionality, and introduces particular requirements. The key factor for an optimized transmission is to efficiently adjust these ingredients in a way, that all components work closely together. The transmission shall provide a network resource adjusted information exchange, while respecting given requirements. In case the requirements and the environment have been identified, an appropriate model must be developed that precisely reflects the corresponding parameters. This model is eventually processed by a suitable algorithm or mechanism, that manages the open tasks and provides a set of parameter that can be used to set up transport protocols, network infrastructure, and the applications.

As multimedia data represents a highly demanding application type, it incorporates a large set of different components, that must be arranged in the optimization processing chain. The main goal of multimedia transport optimization is to reach certain *Quality of Service (QoS) [7]* standards, that have been introduced by the infrastructure, the applications, or the end user. In either case the individual transmission characteristics must be considered.

Multimedia content has an *intrinsic* character, that is introduced by the information source and must be considered for the transmission. This implies that the information contains a predefined requirement for a continuous and timely delivery. For example, if the data source generates content at a certain rate, the corresponding sink must decode the received content at the same rate. Otherwise, the transmitted data may lose information and is finally skipped. In case of an audio transmission, the playback device must be able to proceed incoming data as fast as the sender, in order to avoid audible

artifacts. Accordingly, the real-time transport of intrinsic data represents a remarkable task.

An *elastic* transmission describes the case where no strictly defined transmission rate exists. The information is usable also if it is significantly delayed, paused, or relayed without regarding delivery rates and times. For example, a common file download represents an elastic transmission, as the file contains all information in case the downloading process takes longer than expected.

Multimedia applications generate a high data load for networks as they typically contain high quality content. Video content with a resolution of $1920x1080$ and 64 pictures per second, that is encoded with the H.264 codec with profile *High*, theoretically requires a data rate of 62.5 $Mbit/s$ [8]. In case error correction schemes are applied this data rate is further increasing. The number of multimedia streams in global network infrastructures is continuously increasing. It becomes necessary to handle this traffic carefully in order to avoid congestion situations, as these would lead to uncontrolled packet loss. Hence, multimedia content demands for network infrastructures that are able to manage high data rates while considering special requirements.

Multimedia content is rarely sent to only one receiver. Popular events are typically broadcast in real-time to a large number of receivers. Multimedia is supposed to demand for a one-to-many distribution scheme rather than many one-to-one schemes, as this is considered to be highly inefficient in terms of network utilization. The receiving users have special expectations to the delivered content that must be noted during the transmission. It is assumed that the data arrives timely, in order, and does not exceed a specific residual loss rate, independent of the transmission channel. End users demand for a constant quality, also if they receive content via mobile links. Otherwise the user's *Quality of Experience (QoE)* [9] decreases significantly.

Multimedia content must be handled more sensitive by the network infrastructures and the participating transport protocols. Its characteristics essentially differ from attributes of robust content types as file download or emails.

The amount of multimedia traffic is continuously increasing, and consequently a set of well-known mechanisms and technologies has been evolved. Each main component in the domain of network transmissions is involved. There are special network infrastructure approaches and mechanisms that target to optimize the transport of multimedia content. Special transport protocols have been developed and defined that allow a more efficient data delivery, and applications try to use the network in a way that their content is reliably sent through it.

The task of optimizing data traffic for higher transport efficiency is generally called *Traffic Engineering (TE)*. Basic TE approaches include statistic measures that are achieved by long-term observations, as well as complex simulation results, and best-practices.

The main focus is always on the establishment of a reliable service that can be provided at minimum *cost*. The term cost represents various objectives, such as consumed data rate, required latency, number of hops in the network, or the packet loss probability. *Cost* is a generic term that must be replaced with a situation-dependent metric definition.

Traffic Engineering includes two aspects, as the control is either application-based or network-based. In case the application decides about the data priority, the network must trust the application that the right priority is assigned to the packets. The other case describes the situation where the application completely relies on the decision taken solely by the network. In either case one component must trust the other, but both hardly work together. Corresponding packet classification schemes are *Differentiated Services (DiffServ)* [10] and *Integrated Services (IntServ)* [11]. The packets are marked with an individual priority and the network handles the reliable transport. *Content-aware Traffic Engineering (CaTE)* is an interesting approach that shows how content and infrastructure providers can jointly take advantage of the deployed infrastructures [12]. In [13] Wang et al. present an overview of routing optimization schemes for Internet traffic engineering. Thereby, the authors consider intra- and inter-domain traffic engineering and discuss a set of TE classifications.

Whereas TE focuses on a system-wide approach to find an acceptable traffic distribution, the infrastructure's topology also contains a large potential for multimedia traffic optimization. In case the topology information is known, it is possible to avoid network segments with significant weaknesses regarding the application requirements.

Graph theory [14] represents a viable approach for traffic engineering purposes, that is based on infrastructure's topology information, while abstracting not required side information. It naturally represents the network environment by *edges* and *nodes*, while providing highly sophisticated algorithms to analyze the given topology.

Routing protocols, such as *Open Shortest Path First (OSPF)* [15], use the well-known *Dijkstra's Algorithm* [16] from graph theory to find routes with special properties in the networks. In principle, graph theory can be used to solve multiple tasks in traffic engineering and data distribution, such as one-to-many transmissions. This type of distribution scheme is known as *Multicast* and it uses tree structures to distribute data in the network. Multicast trees can be represented by *Steiner Trees* [17], that can be generated with tools provided by graph theory. Graph theory also provides multiple node metrics, such as degree or closeness, that help to analyze the infrastructure's topology.

Multi-Protocol Label Switching (MPLS) [18] is another technique to control network traffic in packet-switched networks. MPLS categorizes streams of packets and informs routers to equally handle all packets of the same category. This breaks with the common best-effort principle at the network level, as it establishes a circuit-switched end-to-end controlled transmission path. MPLS does not consider in-flow priority differences and thus creates individual transmission channels, in which data packets are handled following the best-effort principle.

MPLS creates an overlay network on top of the real physical network infrastructure. It uses labels to mark packets in order to identify the corresponding path. As these labels represent the categories, they must be network-widely distributed. This can be automatized by the *Resource Reservation Protocol - Traffic Engineering (RSVP-TE)* [19, 20]. It implements a label distribution mechanism to ease the establishment of MPLS networks. MPLS provides an adequate mechanism to deterministically forward multimedia packets in a network, that remarkably reduces packet jitter and the transmission delay. This technique requires administrative access to the network infrastructure, as the nodes must be informed about the labels and their implications. Hence, MPLS is mainly qualified for large managed infrastructures.

New network technologies are considered to contain a high potential to improve the future of data transmissions. As an example, *Software Defined Networking (SDN)* [21, 22] and *Network Function Virtualization (NFV)* [23, 24] describe concepts that aim to further ease transmission handling in large networks. The SDN approach separates the control from data network plane, and consequently establishes two highly specialized operational domains. The control plane contains all required logic to control and manage the network, whereas the data plane is solely responsible for performing actions announced by the control plane. NFV describes a fundamental approach to build traditional hardware-based network functionality in software. This accordingly leads to highly flexible and interactive network applications, as essential services do not longer require specialized hardware. Thereby, individual services are performed by software running on a special operation system, while connected end hosts do not recognize this fundamental new network infrastructure concept.

Transport protocols crucially influence network transport of multimedia content. They may be designed for general purpose data transport without any transmission requirement, or highly specialized to the transport of a certain kind of data. Some transport protocols are accompanied by control and monitoring protocols, that enable a remote management of the sender. Transport protocols have generally different characteristics, so that some kinds of data can be transmitted more appropriate than others. In either case, the applications decide how the data is transmitted to the sink and the network

infrastructure handles the data stream. It is possible that the network guesses the content of a transport protocol via its header information, i.e. the destination port, and starts to prioritize the ongoing transmission accordingly. The identification of multimedia data and its requirements by the help of the packet's header information is not reliable. Chapter 3 introduces multimedia protocols and their special demands in more detail.

Related Research

In [25] Resende and Pardalos give an extensive survey about recent theoretical optimization approaches, which are frequently used in telecommunications. In the following, only a representative subset of all covered topics is considered. The concept of *Integer Programming (IP)* for telecommunications is explicitly introduced by Lee and Lewis [25, chap. 3], and represents a basis for ongoing research in this domain. The authors particularly present an overview of IP models, examples, and solution methods for network structures. In [25, chap. 11] Forsgren and Prytz present an overview of *Telecommunications Network Design*. Performance criteria and the optimization of network design problems, as well as a mixed-integer linear programming solution strategy, are presented in detail and give an excellent overview of this topic. In [25, chap. 17] Klincewicz considers *Quality of Service (QoS)* aspects in the context of network design. The author presents multiple QoS aspects, modeling techniques for delay considerations, and different routing approaches. In [25, chap. 24], Rexford discusses the modeling and the computational challenges of optimizing the tunable parameters of IP networks. In this chapter, network routing in a single network and in large backbone networks is considered, as well as inter-domain routing policies.

In [26], Kachan, Siemens, and Shuvalov compare commercial fast data transport approaches in 10 Gbit/s wide area networks with large latencies and varying packet loss rates. In this work, it turned out that in case of loss, a significant decrease of the TCP transmission data rate down to 1 % of the link's capacity is possible. In summary, these approaches work well in loss-free environments, but they are not suitable for high speed transmissions in error-prone networks.

A pragmatic approach for multimedia transmissions through legacy networks is *Dynamic Adaptive Streaming over HTTP (DASH)*. In [27], Stockhammer describes the standards and design principles of DASH. It is shown, that DASH addresses weaknesses of progressive downloading and RTP/RTSP [28] media streaming, while exploiting existing technologies, e.g. codecs. DASH stores multimedia content at an HTTP [29] server with

different encodings, and each is split into multiple segments. The receiver is able to access the content with a specific encoding according to its requirements and capabilities. Additionally, it is able to iteratively select different chunks of this content block. The streaming procedure is performed via progressive downloading of partitioned content chunks. DASH supports multiple application scenarios, such as on-demand streaming and Live-TV, especially in terms of mobile devices [30].

In 1999 Acharya presents a Ph.D. thesis that targets the improvement of multimedia communication over wide area networks [31]. It deals with the interception of transmission streams at application layer in order to reduce startup delays and server loads. The presented approach, called *MiddleMan*, utilizes distributed cooperative caches, that act as the original multimedia server.

In [32] Border et al. provide a survey of *Performance Enhancing Proxies (PEP)* intended to mitigate link-related degradations. These focus on the improvement of degraded performances of TCP connections in wide area networks. Following the survey, these proxies should be used in specific environments and circumstances. They cover the whole OSI layer stack, but the most popular PEP application is considered at the transport layer with TCP [3]. With PEPs, it is possible to split TCP connections between two nodes, by terminating a connection received from an end system, and establishing a corresponding connection to the other end system [32]. PEPs include different mechanisms to enhance communications, such as *TCP ACK Handling* or *Local TCP Retransmissions*. In [33], Neale et al. present a system for the bi-directional transfer of data packets over a TCP communications system, that can comprise terrestrial and satellite links. The *Loss Domain Separation (LDS)* principle, as discussed in this thesis, can be seen as a specialized ARQ-based PEP for multimedia content in a time constraint environment.

Challenges

There are multiple challenges considering multimedia traffic optimization. In general, the involved components do not interact in a precise way, i.e. they are unaware of the other's requirement. Even in case information is exchanged, the knowledge is abstracted to a group of traffic, rather than to an individual stream or flow. This leads to a limited setup for the optimization process, including the transport protocols, network infrastructure, and the applications.

Transport protocols typically coarsely observe the present network environment parameter, as they reduce the network infrastructure between sender and receiver to a single virtual end-to-end segment. Multimedia applications require a reliable, high granular,

and valuable measurement of the network's characteristics. Consequently, the abstraction of a potentially inhomogeneous network environment inevitably leads to less efficient optimization results for multimedia content.

Available techniques are based on legacy architecture standards, and marginally influence the existing infrastructures that have been invented in the 1980's. These techniques have been developed to transport elastic non-multimedia data, and acceptably work for these content types.

Traditionally, networks and applications are seen as different domains that cannot sufficiently interact with each other. A precise information exchange between both domains would lead to an efficient transportation mechanism. As the amount of multimedia traffic is continuously increasing, new techniques and paradigms are required that set up network infrastructures and applications for upcoming challenges.

1.2 Contribution

This thesis focuses on the identification of individual aspects, that represent key factors for optimized packet-switched networks. Thereby, it addresses the domain of constraint multimedia transmission, as they are considered as a fundamental element in the future media Internet.

Evolutionary and *revolutionary* transport approaches are presented. Evolutionary techniques can mostly be seamlessly integrated into default existing network infrastructures. They target the achievement of an enriched media transmission, while avoiding significant technology changes at the end devices. Revolutionary approaches are discussed, that require deeper interactions in legacy environments. They eventually lead to a considerable extension to current network environments, regarding an optimized multimedia transmission.

The main contributions of this thesis are:

- Network optimization with spatial dimension,
- Multi-dimensional network optimization, and
- Proof of concept demonstrators for future media networks.

These topics include a topology-aware error correction principle, a protocol-based information exchange between the network and applications, and the identification of multi-dimensional metric spaces.

The appendix *Related Publications* provides an overview of all publications that have been published during this thesis.

Network Optimization with Spatial Dimension

Traditional network transmission schemes implement an end-to-end based control mechanism, as known from the *User Datagram Protocol (UDP)* [2] and the *Transport Control Protocol (TCP)* [3]. The end-to-end transmission path between a source and a sink is generally highly heterogeneous, as it is composed of multiple different segments. In case of a mobile receiver, the path consists of at least one wireless and various wired physical transmission channels. Each of these channels has its own properties and characteristics. The logical link's characteristics are consequently an inhomogeneous composition of the properties contributed by the underlying physical channels. The logical end-to-end link constitutes a weak representation of the real infrastructural properties.

Packet-switched networks are generally not error-free, especially if they include wireless segments. In any case, the application of error correction schemes is an advantageous concept. Multimedia content is particularly intolerant to large transmission latencies and high packet loss probabilities. Error correction schemes depend on the underlying network in general, and in particular on the segment's parameters. In case the error correction is not efficiently adjusted to these parameters, it cannot acceptably perform. This means that the amount of information, needed to correct occurring errors, is either higher than normally required, or too low. The first case causes a higher data rate in the network and increases network latencies. In the second case, the transported content is not sufficiently protected, and arising errors during the transmission cannot be adequately corrected. In either case, the non-optimized application of error correction schemes to networks, especially end-to-end networks, remains a suboptimal concept. The focus of this thesis is on the utilization of a general inhomogeneous network topology in terms of error correction schemes. Inhomogeneous topologies contain a currently unused potential to significantly reduce the required data rate for transmissions.

In case the legacy end-to-end transmission path is split at designated nodes in the network infrastructure, special *loss domains* appear. Consequently, the presented optimization scheme is called *Loss Domain Separation (LDS)*. With LDS, it becomes possible to individually set up error correction schemes for predetermined network segments, instead of using a single and weak logical end-to-end segment. The global error correction scheme is accordingly divided into a set of multiple schemes, each performing on separate network segments. The precisely tuned schemes can significantly reduce the network utilization.

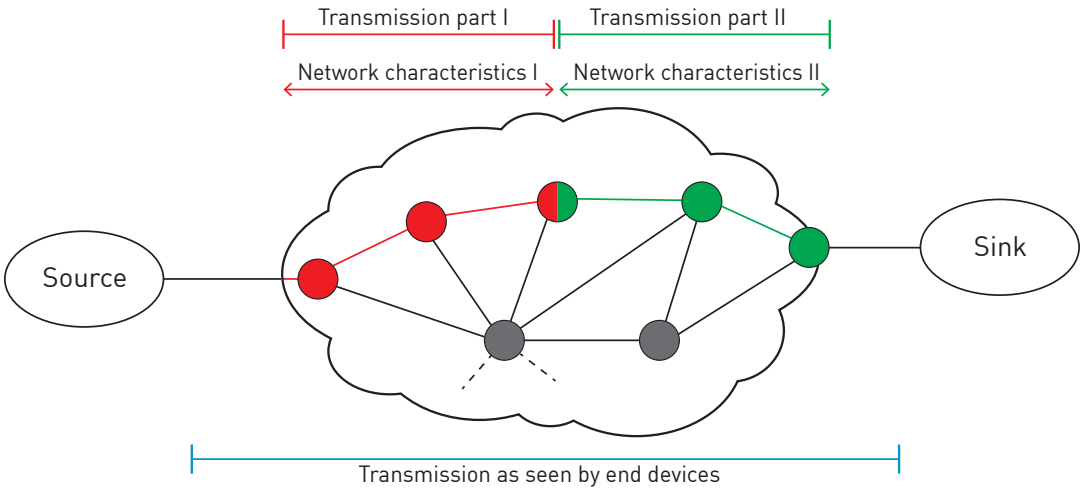

FIGURE 1.1: Optimization approach: Loss Domain Separation.

In case end-to-end constraints must be considered, they represent a global budget. The distribution of these budgets across the available segments is also a sensitive task, as the interdependence between error correction and budget distribution is fundamental for network utilization.

Figure 1.1 presents a schematic overview of the *Loss Domain Separation* principle, by the help of a network that is segmented into two parts.

In this thesis, the application of LDS with different *Automatic Repeat Request (ARQ)* [34] error correction schemes is discussed. The LDS principle follows the *Performance Enhancing Proxies* mechanisms as described in [32].

- **Generic error correction** schemes are discussed without any constraints. The calculations are based on the *Noisy-Channel Coding Theorem* [35], that is valid for any time-unrestricted error correction scheme. It emerged that the application of atomic error correction on a per segment basis outperforms the default end-to-end scheme.

- **Time constraint ARQ and hybrid ARQ** schemes are discussed for transmission environments with a strict time-constraint and an upper limit for the residual error rate. The ARQ calculations are based on a scheme, that allows to flexibly distribute retransmission packets within a set of given retransmission rounds. The hybrid ARQ calculations are based on the *Adaptive Hybrid Error Correction (AHEC)* framework introduced by Tan [36]. It shows, that it is not possible to arbitrarily segment a transmission path. In case the number of segments exceeds a *saturation point*, the application of atomic error correction schemes leads to a

highly inefficient network utilization compared to a default end-to-end transmission path.

With LDS, the complexity of a transmission moves to the network, and specialized intra-network relays are required to perform the protocol terminations. The presentation of the Loss Domain Separation principle is accompanied with a discussion about the selection of acceptable split locations. It includes multiple approaches that analyze the given network topology, as well as an optimization model, that generates precise relay location suggestions according to a given transport environment. Additionally, a flexible metric approach is presented, that provides a mechanism which generates split suggestions from multiple composed metrics.

Multi-Dimensional Network Optimization

Metrics represent a fundamental basis for network transmissions. They reflect individual network parameters and influence routing decisions. Measurable characteristics of network segments are mapped to comparable values, that allow a strict ordering according to a given policy. These measures are eventually used to rate the quality of the networks and their segments. This enables a deterministic network analysis, as well as the selection of appropriate infrastructure parts, by different algorithms and mechanisms. Legacy networks use one-dimensional metrics to find appropriate network paths. This implies, that these generally rely on a single parameter, i.e. the so called *hop count*, that reflects the number of intermediate nodes between the source and the sink. As networks are complex structures including a large set of different components, a one-dimensional representation by metrics is assumed to be inadequate, as other parameters are not considered. The transport of multimedia content introduces special constraints and requirements. The selection of an acceptable network path, that satisfies all demands, is an essential task and represents a significant aspect of packet-switched networking.

In this thesis, a set of different metrics is presented, that can be used to rate the network's quality based on a multitude of different parameters. Additionally, mathematical programs are presented, that enable an optimized network path selection for unicast and multicast transmission schemes. Furthermore, an interactive and revolutionary mechanism is presented, that allows publishing and negotiation of transmission parameters prior to the media transmission by the help of a specialized protocol. With this concept, information, such as available data rates and fundamental transmission requirements, can be explicitly shared between applications and the network infrastructure. The network calculates a suitable distribution tree and tells the application essential network parameters. The data source decides if it can start the transmission with the announced

parameters. The presented specification includes a negotiation process, as well as a periodical update mechanism that allows to immediately react to varying network infrastructure conditions.

Future Media Networking Demonstrators

During this thesis two proof of concept demonstrators for future media networking have been developed.

The first proof of concept is the *Local Wide Area Network (LWAN)* demonstrator. It represents an intuitive blackbox networking system with interactive management and control functionality. It serves as a first basis for future research in the domain of multi-segment transmission. The LWAN demonstrator implements different transport protocols and enables a basic set up of infrastructural parameters.

The second demonstrator extends the fundamental aspects of the LWAN demonstrator. It is based on *Software Defined Networking (SDN)* and represents a fully virtualized networking testbed, and has been tailored to the research topics of the Telecommunications Lab at Saarland University. It allows a flexible network management and a large number of modification possibilities, that is not available in traditional closed source network environments. Besides the local nodes, the environment has been extended by external nodes located at multiple research and industry partners. The SDN testbed has continuously grown and currently serves as a platform for joint scientific research, testing of network protocols, and has been used in multiple Bachelor and Master theses. Both demonstrators have been presented at multiple academic events.

Chapter 2

Data Networks

Data networks are built to connect devices and enable the exchange of information. The design of these networks varies according to their intended use. The probably most prominent representative is the *Internet* [37], which is based on the *Internet Protocol (IP)* [1]. In 1977 the development of the ISO/OSI layer reference model [38] for network design has been started. It states the fundamental basis for a large set of today's wide and local area networks. Numerous end devices are connected and exchange information of any kind, such as email, files, or live multimedia content. Each information type has different characteristics and demands for an individual handling.

With an increasing number of end devices, the right information distribution scheme becomes highly relevant. Sophisticated routing schemes have been developed and implemented to reliably and efficiently exchange information over large network infrastructures. As network transmissions generally facing packet errors, the application of an error correction scheme is required, that recovers lost information at the data sink. Wireless networks, e.g. WLAN or mobile radio, represent frequently used infrastructures, which show that this is a severe requirement.

In the course of time, new network technologies, approaches, and paradigms have been investigated and are about to move towards practical network infrastructures. *Software Defined Networking (SDN)* [21], *Next Generation Networks (NGN)* [39], and *Network Function Virtualization (NFV)* [23, 24] represent only a subset of emerging paradigms and technologies. These approaches are presented in this chapter. The focus is on the topics mentioned above, and on the end-to-end principle, which has been the basis for all information exchange in the last decades.

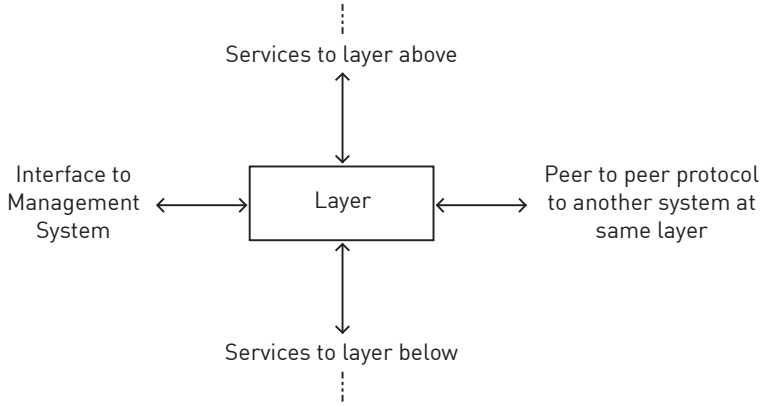

FIGURE 2.1: ISO/OSI layer interaction.

2.1 Open System Interconnection

The *International Telecommunication Union (ITU)*[1] specifies *Open System Interconnection (OSI)* in the recommendation ITU-T X.200 [38] as a model approach, that is used for information exchange between *open systems*. *Open system* means that the end device system complies with the proposed OSI standard in opposition to a proprietary system. It focuses on the interconnection of systems, including transmission of information and distributed task collaboration. The ITU-T X.200 standard provides a set of specifications to ease the communication between multiple autonomous and spatially distributed systems. The concept contains the *abstraction* of functionality into multiple general layer descriptions. It does not provide any specific implementation details. The ISO/OSI reference model is also published as ISO/IEC 7498-1:1994.

2.1.1 Layered Architecture Concept

Layering is the basic mechanism to structure the components of the OSI model. The standard defines a layer as a subsystem that represents an element in a hierarchical system separation. This subsystem groups specific functionality that has a well-defined interface to other layers as presented in figure 2.1. Subsystems can only communicate with subsystems on the directly adjacent layers. For convenience, subsystems and layers are always represented as a vertical stack as shown in figure 2.2. It illustrates the complete OSI models of two end device systems connected via a network relay device, e.g. a router. Information designated to be sent on any layer must always pass the stack downwards whereas received information moves upwards in the stack. Each subsystem may contain multiple elements that represent equal functionality, but are implemented differently. Elements of a certain layer can communicate with other elements on the

[1] http://www.itu.int/en/ITU-T/Pages/default.aspx (2015/01/29)

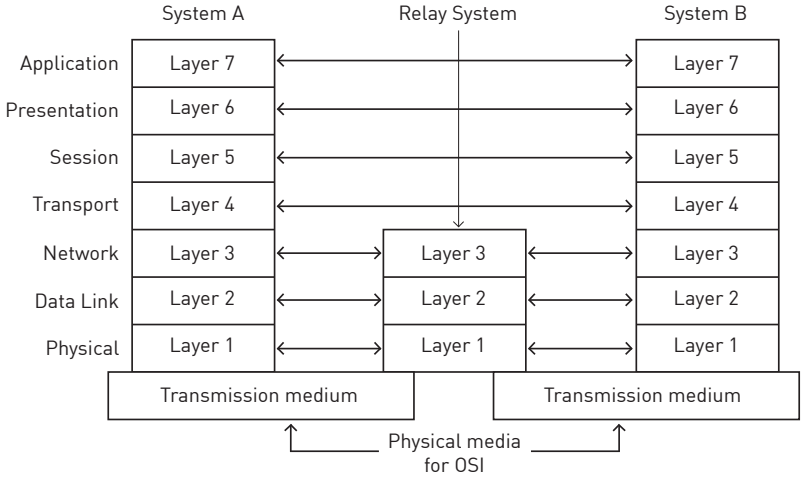

FIGURE 2.2: The ISO/OSI layer model following ITU-T X.200.

same layer, but on a different system. The recommendation of ITU-T X.200 [38] also highlights *Quality of Service (QoS)* [40] handling. Depending on the connection mode it names possible parameters: For a single connection-mode transmission these parameters include *expected transmission delay*, *probability of corruption*, *probability of loss or duplication*, *probability of wrong delivery*, and *protection from unauthorized access* [38].

2.1.2 The Specific OSI Layers

The OSI reference model identifies seven layers, each including specialized functionality for packet communication. Each layer provides a specific set of functionality and is accessible via interfaces located at the borders to its directly adjacent layers. A layer offers information and functionality to the layer above and consumes from the layer below as indicated by figure 2.1.

Application Layer (OSI L7)

The application layer provides the interface for any application to access the *Open System Interconnection Environment (OSIE)* [38]. It represents the topmost layer and includes all functionality to communicate with other OSI systems. Conceptually, this layer does not contain any function provided by lower layers, such as congestion control or signal modulation. This layer contains all functionality required by humans or computer applications. The application layer is responsible for the exchange of information between applications, including the identification of intended communication partners or the selection of an acceptable *Quality of Service* parameter set.

Presentation Layer (OSI L6)

The presentation layer service is described in ITU-T X.216 [41]. The main purpose of this layer is to provide a system independent representation of data. This allows a reliable and deterministic information exchange between systems. The presentation layer is responsible for the negotiation of the transfer syntax and the usage of session services. It provides a set of functionality to the application layer, including the identification and selection of a set of transfer syntaxes as well as access to the session services. This layer is no longer explicitly defined in network scenarios, as it has been merged into the application layer.

Session Layer (OSI L5)

The session layer, as described in ITU-T X.215 [42], provides functionality for a synchronized and reliable exchange of information within sessions. It establishes a session-connection between two presentation layer entities. Besides functionality for session-connection establishment, it also contains exception reporting, and resynchronization features. As with the presentation layer, it is no longer used and has also been merged into the application layer. The *Hypertext Transfer Protocol (HTTP)* [29] is a prominent representative of a protocol that works inside the combination of OSI L5, L6, and L7 with merged layer specifications.

Transport Layer (OSI L4)

The transport layer, as described in ITU-T X.214 [43], enables a transparent transfer of data between session layer entities. It attempts to efficiently utilize the network resources while handling constraints introduced by the session layer or the network service. This layer has an end-to-end characteristic, as a transport layer connection is solely possible between end hosts implementing the OSI model. It does neither perform routing, switching, or relaying action nor does it utilize functionality according to the present network application status. The transport layer provides a transport address to the session layer that can be used to uniquely identify the session entity at the host. The layer contains multiple functionalities, including end-to-end multiplexing of transport connections into network connections, end-to-end error detection, monitoring of QoS parameters, and supervisory functions.

Network Layer (OSI L3)

The network layer, as described in ITU-T X.213 [44], establishes, maintains, and terminates network connections between end hosts implementing the OSI model. It includes functionality to exchange transport layer data on a network connection basis. This layer provides a specified interface to the transport layer in order to keep it independent of the used underlying communication media. It offers a basic service for a transparent transport data exchange while handling a wide variety of networking configurations, including the transport between two end hosts over complex cascades of subnetworks. The network layer includes routing, relaying, sequencing, and flow control mechanisms as well as functionality for error-detection and -recovery.

Data Link Layer (OSI L2)

The data link layer, as described in ITU-T X.212 [45], builds a data link connection over one or more physical connections. It detects and handles errors introduced by the underlying physical layer and enables the network layer to handle interconnections of data circuits within the physical layer. It provides data link addresses to the network layer. This layer includes functionality for error detection, routing, and relaying as well as control of data circuit interconnections based on subnets.

Physical Layer (OSI L1)

The physical layer, as described in ITU-T X.21 [46], represents the establishment and maintenance of physical connections between open systems. It is used by the data link entities and performs bit-wise transmission of information over physical media. As the number of potential physical media is large, this layer is generally supposed to provide physical connections, sequencing, and data circuit identification. Detailed service information of the physical layer depends on the present medium and must be individually specified.

2.2 The End-to-End Principle

The interconnection of devices is generally dedicated to the exchange of information between all connected parties. The participating hosts establish a communication channel, that is defined by the corresponding end points. Data networks are designed to

be highly flexible as they provide the basis for exchanging a plethora of different information types. Current networks cover the transmission of file downloads, emails, voice, and multimedia, such as real-time video streams. Each of these application scenarios has different requirements. As an example, file transfers assume that a file is transmitted reliably, whereas live streaming of multimedia content is generally able to cope with a specified amount of loss. The network does not have any information about the type of information being transmitted, and consequently cannot handle information accordingly.

The end points are responsible for transmission as well as ensuring information integrity, as the complete transmission logic is concentrated at the network's end points. The end-to-end principle exactly states this observation [47]:

> *The function in question can completely and correctly be implemented only with the knowledge and help of the application standing at the end points of the communication system. Therefore, providing that questioned function as a feature of the communication system itself is not possible.*

This description was first stated by Saltzer et al. [47] and still represents one of the most popular communication protocol design guidelines. It is a general design approach that must be carefully reviewed as the meaning of *end points of a communication system* is not invariably defined. Assuming two people talk to each other, the *end points* can be seen as the people themselves. To be more specific, their brains processing the audio input coming from their ears. In case a file transfer between computer devices is considered, it is difficult to precisely define the term, as 1) the file must be read from hard-disk, 2) pushed through the application processing pipeline, 3) sent via a network socket to another device, 4) received from a network socket, 5) pushed to the receiving application. According to the OSI reference model discussed in section 2.1, the network stack represents only an interim stage and not the end point of the communication. Information is forwarded to OSI layer 7 and is potentially further processed afterwards. In general, the end point of a communication system is the last entity in the transmission chain.

It can be seen that the OSI layer model is not absolutely perfect as the network layer does not terminate the transmission chain in the common sense. Information must be passed through multiple layers where specific functions are invoked prior to the reception by the end user or entity. Interestingly, the majority of applications assume that the error probability in the end system equals zero. As an example, the majority of designers assumes that there are no errors when reading data from the main memory. Indeed, errors may occur and distort application communication.

The end-to-end principle implicitly assumes a neutral network. The complete transport intelligence resides at the network edges represented by end points. In chapter 5 an optimization approach is presented that splits end-to-end connections in order to improve the transmission efficiency. The classic end-to-end principle is abolished and replaced by a sophisticated transparent relaying mechanism. The presented approach is rather seen as an evolutionary extension to the end-to-end principle than a violation, as it focuses on the special handling of multimedia transmissions with strict constraints.

2.3 Network Characteristics

Network characteristics describe basic parameters and measurements influencing transmission and its efficiency. The topology of the network infrastructure simultaneously represents an individual performance indicator and a constraint for transmissions. The way nodes are interconnected is a crucial aspect for data transmissions, as it affects domains such as reliability, latency, and resource utilization. Besides the main appearance of the network, its intrinsic behaviour is highly significant, as it unveils transmission aspects concerning latency, data rate, loss probability, or jitter.

Another characteristic is the transmission scheme itself. Traditional networks are built according to the end-to-end principle. This creates a virtual link between the communication end points and merges their parameters of the segments in between. In terms of transmission efficiency, this leaves significant room for improvements. A potential improvement, called *Loss Domain Separation*, is discussed in chapter 5. Connection establishment can use circuit- or packet-switching, which differ in their parameter set as well as the used physical media. In the following, an overview of the most significant network characteristics is presented.

2.3.1 Topology

The network topology describes the nodes and their interconnections, thus it represents the network structure. Topologies can be separated into two main groups: *physical* and *logical* topologies. The *physical* topology reflects the physical node locations and the connections between nodes. As an example it describes the physical deployment of routers, switches, and other network equipment. The *logical* topology represents a virtual overlay network on top of the physical topology. Peer-to-peer [48] or other overlay networks [40] are prominent representatives for logical infrastructures. Multiple different topologies, each with its individual advantages, can be identified. Figure 2.3 illustrates the most prominent topology types for IP networks.

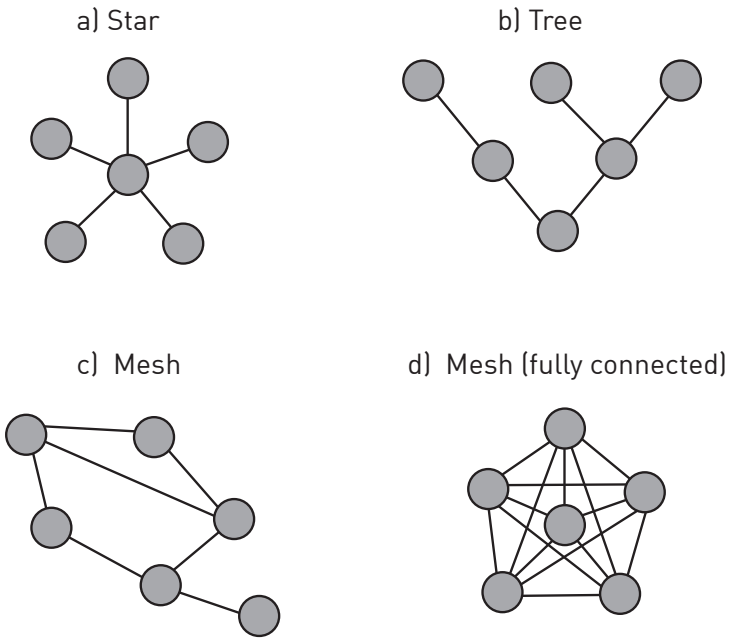

FIGURE 2.3: Examples of a *Star*, *Tree*, and *Mesh* topology.

In legacy IP networks each node contains full logic and operates independently. The nodes have computational functionality to analyze incoming and compose outgoing information. Routing and switching decisions are separately made by each and every node in the network. Traditional networks have a decentralized information processing and distribution behavior where no node is prioritized. The network devices exchange information via different protocols. For example, routers employ *routing protocols*, such as *Border Gateway Protocol (BGP)* [49], *Intermediate System to Intermediate System (IS-IS)* [50], or *Open Shortest Path First (OSPF)* [15]. The totality of all information transposed by routing protocols enables each participating router to maintain a topology representation. Network topologies are generally represented and analyzed by tools derived from graph theory [14]. These tools allow routing between nodes or the selection of distribution structures. The most prominent mechanisms in this domain are *Shortest Path* [16] or *Spanning Tree* [16] algorithms. Graph theory additionally supports the analysis of node connectivity, e.g. by the help of ingress and egress *degree* [51].

Upcoming technologies, such as *Software Defined Networks (SDN)* [21] or *Network Function Virtualization (NFV)* [23, 24] tend to dissolve this traditional topology structure and set up new node interaction and transmission approaches. Further information about SDN and NVF are presented in section 2.7 and 2.8.

2.3.2 Performance

The network performance reflects the intrinsic quality of a network infrastructure and depends on multiple characteristics.

Transmission data rate is a value that provides information about the ability of the whole network or a segment to transport data in a specific time interval. It is typically measured in *bits/sec*. A packet with a fixed size requires less time to be sent with a higher data rate compared to a lower data rate. The term *data rate* is often confused with *bandwidth*. In the telecommunication domain bandwidth represents the spectral bandwidth of a signal with the unit *Hz*.

The *transmission latency* represents the time to deliver a packet from one node to another. In case the nodes are directly connected by a single link, it refers to the links latency. If multiple links are used to connect the nodes, the term reflects the path latency. It consists of three ingredients: the propagation delay[2], the packet *transmission delay*, and a potential *queuing delay*. Section 3.2 further discusses the transmission latency.

Packet jitter is closely related to the transmission latency. It represents the time variation due to the network load encountered by subsequent packets. Consequently, networks with fixed transmission latency do not introduce any packet jitter.

Another key characteristic of network performance is *packet loss*. It describes the situation when data packets are not received by the sink. A *packet error* is related to packet loss. For example, a packet error represents the fact that received packets contain defective bits. In case of a transmission over an erasure channel [52] a packet error is equal to a packet loss, as all packets reaching the receiver are assumed to be error-free. If one bit in a packet is not valid, the whole packet is dropped. A typical situation facing packet loss is owing to physical distortions, as introduced by wireless and radio environments. Error correction schemes help to overcome this challenge by proactively adding redundancy to the transmitted data at the source. This redundancy is used at the sink to recover distorted packet information. Packet loss additionally originates from *congestion*. In this case, the network devices cannot buffer any more incoming packets, which are then dropped. Section 2.5 provides more information about error correction schemes and their application. Packet loss caused by physical phenomena can be handled by application layer mechanisms, whereas loss owing to network congestion is dealt within *traffic engineering* and requires network wide interactions. The aforementioned terms can either be used for virtual and physical networks as well as individual segments.

[2]The propagation delay depends on the propagation medium.

A more statistical measure for network performance refers to *reliability*, *maintainability*, and *availability (RMA)* [53] of the network. *Reliability* reflects the rate of failures of the complete network or of one of its components. *Maintainability* represents the amount of *effort* required to restore or upgrade the network or a component after a failure occurred. *Availability* means the relation between *reliability* and *time of recovering* the environment. *RMA* cannot be directly measured, but achieved via policies and metrics. The network performance significantly influences the *Quality of Service (QoS)* [40] and *Quality of Experience (QoE)* [9].

2.3.3 Transmission Design

Traditional networks are designed as *end-to-end* networks that implement the previously mentioned principles. In this case, payload data is transferred between the source and the sink and the transmission is fully controlled by the transmission end points. The packet header is inspected at each intermediate node and the packet is individually forwarded to a network device located closer to the packet's destination. Besides this forwarding, no actions are performed at the intermediate network components, and the packet is originally moved to the next device.

The connection can be consequently seen as a single virtual link between source and sink. The virtual link reflects merged properties, such as latency and packet loss, according to the participating network segments. The end points cannot distinguish at which point a packet faces high latencies or is dropped. The application must always consider the virtual link and thus the network path as a whole. With end-to-end architectures it is not possible to individually apply error correction schemes that are specific to a given network segment, and efficiently utilize available resources.

A *multi-hop* approach employs information about individual network segments. This allows the application of specific networking technologies and parameters for each path segment.

In case of multiple segments with significantly differing parameters, such as latencies, this leads to the application of an individually tailored error correction scheme for each segment. As an example, consider two lossy network segments where the round trip time of segment `1` is significantly smaller than the round trip time of segment `2`. In this case, a retransmission scheme is suitable solely for segment `1` whereas segment `2` should use a forward error correction scheme [34, 54]. This concept frees network segments from overhead traffic generated by error correction schemes on other segments. Consequently, the available network resources are used more efficiently. Multi-hop transport schemes lead to a *spatial separation* of network segments in order to individually handle network

conditions. This approach requires more knowledge about the network as well as sophisticated network technologies, as data streams must be terminated, processed, and reestablished inside the network, without any participation of the end points. Chapter 5 introduces a *loss domain separation* scheme that enables a segment specific error correction in order to efficiently transmit multimedia content.

Network transmissions use a communication channel that connects source and sink. Multiple different channel-establishing concepts are known, each with its individual implications to a network transmission. *Circuit-switching* [40] implies an exclusive and fixed communication channel establishment before the actual information exchange. This also includes resource reservation along the selected network path. This ensures a guaranteed data rate and transmission latency, and lowers the probability of packet jitter close to zero. This connection type creates a virtual link between the end devices and all network packets take the same route through the network.

In contrast, *packet-switching* implies that each packet is individually handled at each node in the network. There are no guarantees for any properties, such as data rate, latency, or the route through the network. Each packet of a data stream can potentially take different ways through the network, resulting in different arrival times at the receiver. This causes packet reordering and introduces jitter. This concept is often named *best-effort* routing.

Technologies like *Multi-Label Protocol Switching (MPLS)* [18] focus on the implementation of circuit-switched network connections in packet-switched environments. Basically, MPLS uses IP packets and their address space, but allows routers to analyze a small flag in the packet header to decide where to forward the packet. Therefore, each router must maintain a specific table including flags and designated forwarding ports. This ensures a reliable and replicable packet delivery over all MPLS enabled routers. Section 3.4 describes the most prominent multimedia transport protocols in more detail.

Another transmission design aspect is the mode how data is transmitted inside the network. IP-based [1] networks split information into small data chunks of variable size before transmission, which are typically smaller than 1500 bytes. These data chunks are prefixed with a packet header that contains all relevant transmission information. Another approach is to use fixed size cells as implemented in *Asynchronous Transfer Mode (ATM)* [40] networks. The small data chunk size with ATM allows a fast transmission over networks. The corresponding data carrying efficiency, i.e. the header to payload ratio, is $^{5}/_{48} \approx 0.10$ whereas the ratio with IP is $^{60}/_{1500} \approx 0.04$ [1, 55].

2.4 Routing Schemes

Routing deals with the analysis of the network topology and the selection of suitable paths in the network that reliably connect source and sink. The concrete implementation of this depends on the network and the transmission. A one-to-one data exchange requires an adequate network *path*, whereas a one-to-many connection requires a distribution *tree*. IP-based networks are generally best-effort routing networks. They individually select the transmission path in the network for each packet and do not provide or exploit any information about the global efficiency of routes in the network. Consequently, they utilize only a local view of the network.

There are situations where it is sufficient to find the *best path* or the *best tree* in the network. *Best* in this context implies the existence of a path whose parameters are optimal and that there exists no other path with better characteristics. *Metrics* and *norms* are used to measure and judge the quality of a network link or path. Section 4.4 introduces metrics and norms in a detailed manner. Metrics reflect the underlying network parameters, such as latency, packet loss rate, or jitter. Figure 2.4 presents the most prominent routing schemes that are also discussed in the following. Routes are often calculated based on optimization and graph theoretical [14] algorithms, which are presented in sections 4.2.1 and 4.3.

2.4.1 Unicast

The transmission between a single source and sink is called a *unicast* or *one-to-one* connection. Devices are identified by their IP addresses. In the domain of graph theory, unicast routing implies the selection of a single-source shortest path in the network, e.g. by the application of *Dijkstra's Algorithm* [16] or the *Bellman-Ford Algorithm* [16]. Unicast routing is practically performed by the *Routing Information Protocol (RIP)* [56] and *Open Shortest Path First (OSPF)* [15].

2.4.2 Multicast

Multicast allows an efficient data exchange between a single source and a set of receivers, which is also called a *one-to-many* connection. Multicast introduces the concept of *groups*, where each group contains multiple receivers and is uniquely identified by a special multicast IP address. The source sends data to a set of receivers identified by a multicast group. It addresses the packets to the corresponding multicast address and the network manages all required routing operation based on the multicast address. The

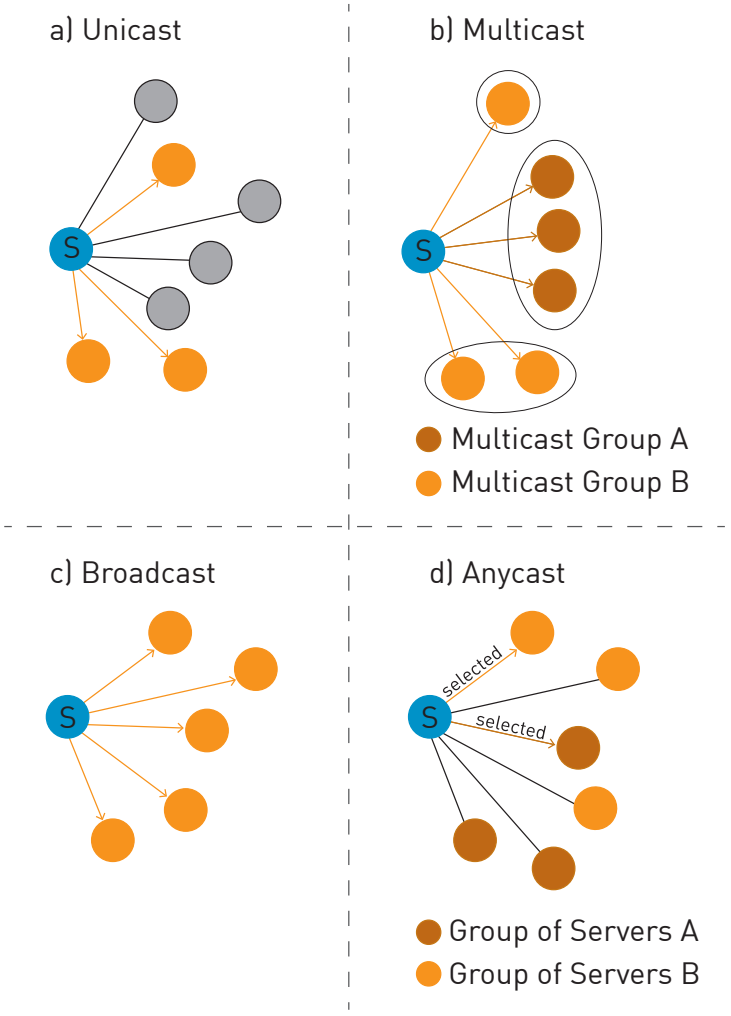

FIGURE 2.4: Routing scheme overview.

source establishes a single unicast connection to the multicast address whereas the network handles all remaining tasks. In case a single server sends the same data to multiple receivers, multicast uses network resources more efficiently than unicast, as it avoids the establishment of individual overlapping connections between the sender and each sink. In the domain of graph theory, multicast is implemented by finding a suitable tree structure containing a subset of all available nodes. A structure with these characteristics is called *Steiner Tree* [17]. These are computationally complex and can be found using approximation algorithms [57]. In practice, multicast distribution trees are calculated by protocols such as *Distance Vector Multicast Routing Protocol (DVMRP)* [58], *Multicast Open Shortest Path First (MOSPF)* [59], or *Protocol Independent Multicast (PIM)* [60, 61].

2.4.3 Broadcast

Broadcast is a special case of multicast where *all* sinks in the network are addressed by a single sender. In analogy to multicast, the source sends packets to a special address and the network handles the succeeding packet distribution. Broadcasting uses special broadcast addresses from the network's IP address space. In a graph theoretical context, broadcast routing leads to the selection of *Spanning Trees* [16]. These trees can be obtained by algorithms from *Kruskual* and *Prim* [16].

2.4.4 Anycast

This approach targets the case where a service is hosted on a group of servers. In case a device is going to send data to such a service, it cannot unambiguously identify an individual server where to send the request. Anycast allows a device to send packets to a special anycast address and let the network handle the data distribution to at least one server of the group. This concept eases the search and selection process for a special server offering an individual service (e.g. Content Deliver Network (CDN) [62] mirrors). It significantly decreases latencies and enables load balancing policies. This routing scheme can be modeled as a special case of the graph theoretical *All-Pairs Shortest Path (APSP)* problem [16]. The *APSP* can be solved by the *Floyd-Warshall Algorithm* [16]. A central instance, having knowledge about all paths in the network, can provide information for routing mechanisms to select appropriate anycast routes for an individual pair of source and sink. Further information about *Anycast* is given in [63].

New network technologies, such as *Software Defined Networking (SDN)* [21], which is presented in section 2.7, are predestined to further implement new and improve legacy mechanisms, such as unicast, multicast, broadcast, or anycast.

2.5 Error Correction Schemes

Transmitting data over any physical or logical transmission channel incorporates the risk of transmission errors. *Error* implies that the received data at the sink is different to the sent data at the source. Transmission errors can have different causes, such as physical channel distortions or congestion effects. Errors inherently disturb the communication between the sender and receiver. Consider a video stream where the transmission is distorted, so that the receiver is not able to properly display the video, leading to an

overall decrease in *Quality of Service* [40] and *Quality of Experience* [9]. It is necessary to protect the communication with error correction schemes.

Redundancy information is used by the receiver to recover the original data packets in case the received packets contain errors. In case a transmission error occurs, the receiver uses this redundant information to reassemble the original data packets. If the amount of redundancy is sufficient, the data stream can be perfectly reassembled at the receiver. Otherwise some residual errors remain. There is no guarantee that the original information can be recovered perfectly.

The two most prominent categories of error correction techniques are the *retransmission of corrupted packets* and the *insertion of additional data* to handle potential information loss. The data sink must be able to identify the error position in order to start the recovery process. The according mechanisms are called *Automatic Repeat Request (ARQ)* [34] and *Forward Error Correction (FEC)* [54], whereas the combination of both schemes is called *Hybrid Automatic Repeat Request (HARQ)* [36]. In the following these three error correction approaches and their applications in multi-hop networks are discussed.

2.5.1 Automatic Repeat Request

With *Automatic Repeat Requests (ARQ)* [34] the receiver is able to detect and reorder corrupted packets within a data stream. The packets contain a sequence number that enables the receiver to determine a missing packet. Lost packets are retransmitted out-of-order from the source. This reordering mechanism requires a *bidirectional* transmission channel, as the sink requests missing packets from the source. ARQ describes a *reactive* error correction scheme.

The *Automatic Repeat Request* can be implemented in three variants:

- *Stop and Wait:* This is the most naive mode, where the sender transmits a packet and waits for the receiver's *acknowledgement (ACK)* of a successful packet transmission. The next packet is only sent once the ACK of the prior packet has been properly received.
- *Go-Back-N:* In this variation, the sender also waits for an ACK and sends a number of packets without waiting for an acknowledgement. The actual number of packets in transmission is determined by a *window*. If the window has a size of N packets, the sender can send up to *N-1* packets after the first was sent. The source waits for an acknowledgement of the sent block and accordingly shifts the window *N-1* slots further.

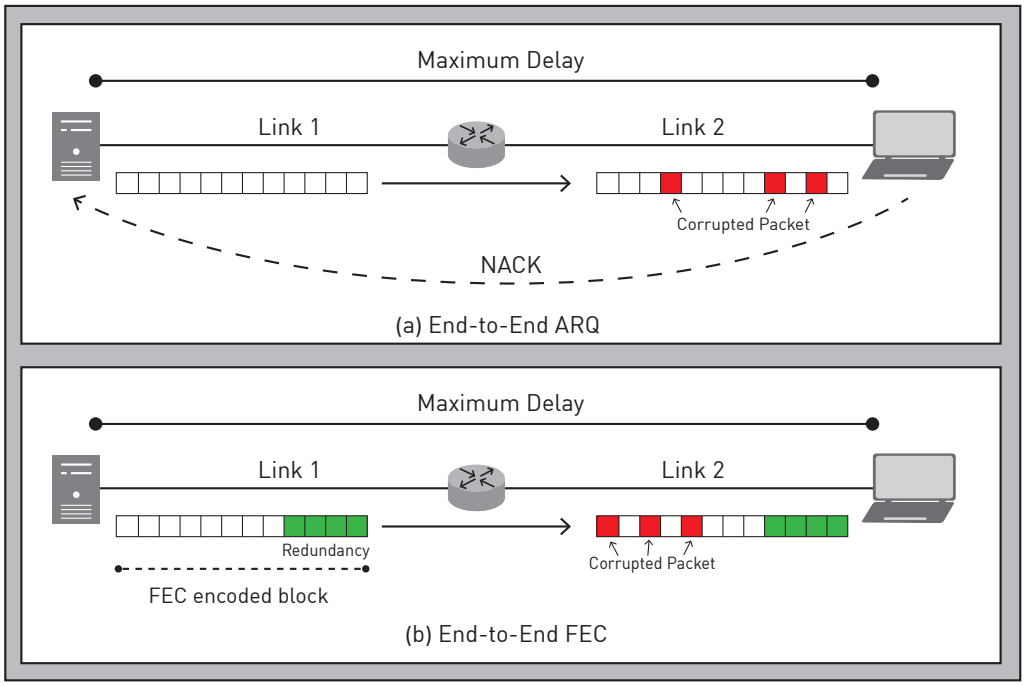

FIGURE 2.5: End-to-end error correction schemes.

- *Selective-ARQ:* In this mode, the sender does not wait for an ACK that the packets have been received in good order. It listens for *negative acknowledgements (NACK)*. In case a NACK is received, the requested packet is individually resent from the source.

With an ARQ scheme the receiver optimally experiences the same quality all the time. The data rate on the communication channel may significantly vary over time, depending on the network status and occurring errors. ARQ is a time intensive error correction scheme, as missing or corrupted packets are re-requested by the receiver. This results in a latency of at least one *Round Trip Time (RTT)* [64]. RFC 3366 [34] provides a general overview of ARQ techniques. The application of an end-to-end ARQ scheme is illustrated in figure 2.5 a).

2.5.2 Forward Error Correction

Forward Error Correction (FEC) [54] is a *proactive* broadcasting error correction scheme. It does not require bidirectional communication between the sender and receiver, as the source continuously pushes packets with added *redundancy* towards the receiver. This is independent of the current network status. FEC allows a constant data rate over the time of transmission, whereas the quality may vary, depending on the number of errors occurring on the transmission channel. *Forward Error Correction* can be implemented

for a transmission requiring nearly fixed latencies, as it represents a unidirectional scheme without retransmissions over large network parts. Prominent examples of FEC are *Hamming* [35] and *Reed-Solomon* codes [65]. The application of an end-to-end FEC scheme is illustrated in figure 2.5 b).

2.5.3 Hybrid Automatic Repeat Request

Hybrid Automatic Repeat Request (HARQ) [36] is the combination of an *Automatic Repeat Request (ARQ)* and a *Forward Error Correction (FEC)* scheme. The FEC is used to reduce the frequency of retransmissions. The added redundancy enables the sink to proactively correct some transmission errors without requesting any retransmissions. The retransmission scheme ARQ is used to request the missing information from the sender in case the preceding FEC has not been able to correct encountered errors.

HARQ is perfectly suited for large, independent receiver groups, as FEC increases the scalability of the error correction and ARQ introduces adaptivity by adding an on-demand bidirectional communication mechanism.

Hybrid Automatic Repeat Request can be categorized into two groups:

- **Type I HARQ:** ARQ and FEC work consecutively. Data is sent with added redundancy. In case the data is not recoverable at the sink, the sender is requested to retransmit the whole data chunk. Type I HARQ works similar to a pure ARQ.
- **Type II HARQ:** ARQ and FEC closely work together. The sender transmits data without redundancy information. In case the sink is not able to decode a data block, it solely requests redundancy information from the sender in order to recover the original data.

In [36, 66] a hybrid error correction framework has been specified and implemented. It also incorporates adaptive parameter selection for both, ARQ and FEC components, as well as a sophisticated congestion control mechanism. This framework has been developed at the Telecommunications Lab[3] of Saarland University and is called *Predictably Reliable Real-Time Transport (PRRT)* [66]. Section 3.5 presents PRRT in more detail.

[3] http://www.nt.uni-saarland.de/en/projects/running-projects/prrt.html (2015/01/29)

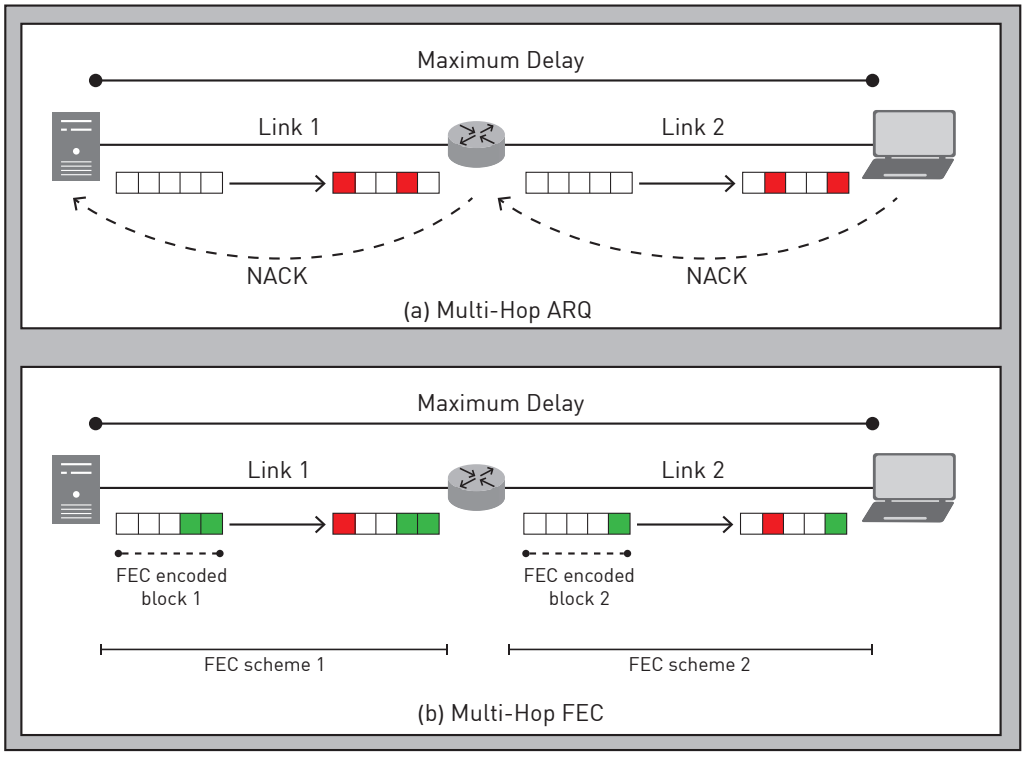

FIGURE 2.6: Multi-hop error correction schemes.

2.5.4 Multi-Hop Error Correction

Multi-hop error correction is applied in multi-hop networks as introduced in section 2.3.3. The logical end-to-end error correction domain between source and sink is split into individual error correction domains, based on individual network segments. The error correction schemes are consequently assigned to particular network path segments and the path is divided into multiple loss domains. Each domain or segment is individually handled considering its characteristics. As an example, FEC is used for segments with larger latencies, and ARQ for segments with shorter latencies, according to their parameter set. Nodes connecting two segments must additionally provide functionality to terminate and re-establish connections, typically up to the OSI network layer (L3). This implies that more logic and complexity have to move towards the network core. Figure 2.6 illustrates a multi-hop application for ARQ and FEC error correction schemes with two domains. Chapter 5 proposes an optimized multi-hop error correction with *Loss Domain Separation*.

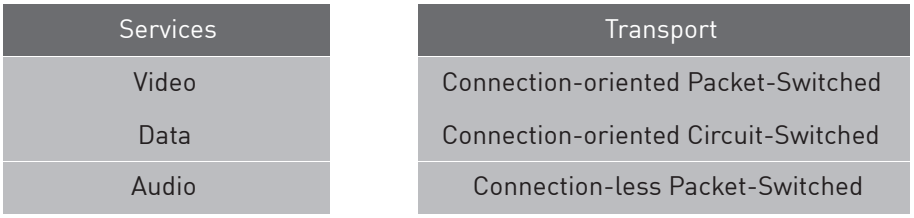

FIGURE 2.7: Next Generation Network layer.

2.6 Next Generation Networks

Traditional IP-based networks are generally implemented according to the seven-layer OSI reference model specified in [38] and discussed in section 2.1. This model has rigid structures, and common networking techniques, such as TCP/IP [3, 1], do not adhere to this structure at all.

Next Generation Networks (NGN) [39] describe a new type of packet-based network technology that replaces traditional circuit-switched telecommunication infrastructures, such as the *Plain Old Telephone Switching (POTS)* system. As many traditional network technologies and infrastructures are still in use, NGNs must be able to interconnect to legacy network types in order to maintain high level communication.

Next Generation Networks represent a reality that describes the ongoing move towards a unified all-IP environment for communication systems.

The telecommunication standardization sector of the ITU (ITU-T) provides a general overview in [67] and proposes a set of general principles as well as a reference model for NGN in [68]. These differ from the OSI reference model as they do not necessarily have seven layers of functionality. The available NGN layers may provide a different functionality with a different focus, i.e. the separation of *services* and *transport*. Figure 2.7 shows both layers and some exemplary services and transport concepts. They enable a separate offering of services and ease the individual evolution of each layer and its functionality. All layers can be further divided into three sublayers: *data*, *control*, and *management*. Each sublayer has its own roles, task domains, and participants. ITU-T Y.2001 [67] defines a *Next Generation Network* as follows:

> *A packet-based network is able to provide telecommunication services and able to make use of multiple broadband, QoS-enabled transport technologies and in which service-related functions are independent from underlying transport related technologies. It enables unfettered access for users to networks and to competing service providers and/or services of their choice. It supports*

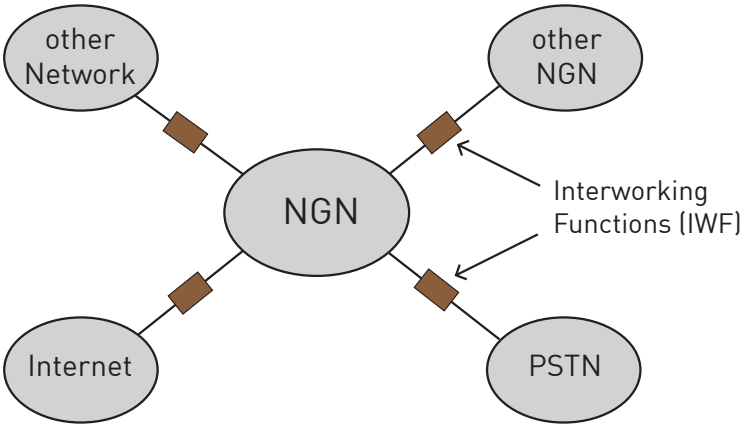

FIGURE 2.8: NGN interconnections with legacy networks.

> *generalized mobility which will allow consistent and ubiquitous provision of services to users.*

The *service layer* manages the transfer and control of service-related data. It handles service resources and network functionality for users and applications. Due to the large number of different applications this layer is coarsely described and represents a general specification. Application's demands may be specialized, so that they require the implementation of different services with individual functionality.

The service layer follows the recommendations specified in ITU-T G.809 [69].

The *transport layer* provides functionality to transmit bit-oriented data between end devices. This layer represents a cascade and recursion of multiple different network types. It provides connectivity and includes transport related functions. The transport layer can be set up by different technologies as packet-switched, circuit-switched as well as connection-oriented or connection-less communication channels. As specified in the recommendation ITU-T Y.2001, *Next Generation Networks (NGN)* are packet-based and prefer the *Internet Protocol (IP)* [1] as their main transport protocol. It also allows the connection of NGNs to legacy networks via *Interworking Functions (IWF)* as shown in figure 2.8. The transport layer follows the recommendations specified in ITU-T G.805 [70].

A fundamental characteristic of *Next Generation Networks* is their explicit support for multimedia services, such as audio or video applications. The ITU-T recommendation Y.2001 [67] does not limit the flexibility of NGNs by defining special user access schemes or types of protocols.

Next Generation Networks represent a new approach for network technologies and infrastructures. As the default network environment basically consists of traditional IP

FIGURE 2.9: NGN layer communication via Interworking Functions (IWF).

networks that follow the OSI reference model, it is required to seamlessly connect NGN and non-NGN environments. This connectivity ensures a gradual deployment of NGN and enables access to services residing in legacy networks. NGNs consist of two individual layers, one for transport and one for service purposes. In either case the corresponding layers of two different networks are connected via *Interworking Functions (IWF)* implemented on *Interworking Units (IWU)* as proposed in ITU-T Y.1251 [71] and illustrated in figure 2.9. IWFs are layer specific and unique as they differ among the layers.

Besides the ITU-T recommendation for *Next Generation Networks*, the *European Telecommunications Standards Institute (ETSI)*[4] also specified an NGN in its *Telecommunications and Internet converged Services and Protocols for Advanced Networking (TISPAN)*[5] workgroup. The specification can be found in ETSI ES 282 001 [39] and is based on the IP Multimedia Subsystem (IMS) [72]. The *3rd Generation Partnership Project (3GPP)*[6] additionally works on NGN architectures based on the IMS but focuses on mobile technologies.

Software Defined Networking (SDN) [21] represents a concrete approach of the *Next Generation Network* definition as it separates the control and data plane of a network. This aligns with the NGN's abstract API for sending packets and requesting features from the network. SDN additionally features a prominent open interface protocol called *OpenFlow* [22].

2.7 Software Defined Networks

Software Defined Networking (SDN) is a recent approach to ease the management of distributed networking devices by enforcing centralization. A special entity in the network gains global knowledge about the topology and active transmissions. It uses a standardized protocol for information exchange with the network infrastructure devices. This entity is called *controller*.

The network intelligence moves from individual network devices to the controller. It takes decisions and forwards operation instructions to the network devices, such as

[4] http://www.etsi.org/ (2015/01/29)
[5] http://www.etsi.org/technologies-clusters/technologies/next-generation-networks (2015/01/29)
[6] http://www.3gpp.org (2015/01/29)

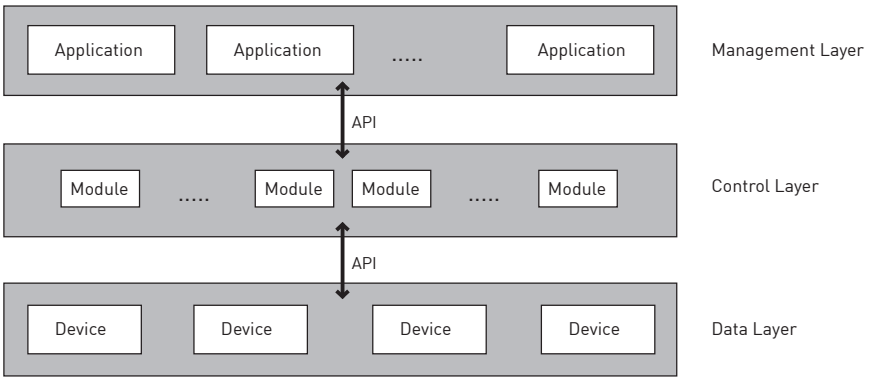

FIGURE 2.10: Software Defined Networking architecture overview.

forward incoming packets to port X or *drop all ICMP packets*. This concept leads to a strict separation of workflows into a *control plane* and *data plane* as illustrated in figure 2.10.

The controller, given its global knowledge, enables a more efficient way to compute network paths for streams. For example, it is able to consider potential congestion and lossy segments, as well as to review the network status including data rates. It communicates with network devices using a standardized protocol, which interacts with network hardware of different vendors and leads to a fast configuration and maintenance procedure. *Software Defined Networking* also enables a new kind of automation, as the controller is able to handle configuration failures or network-wide configuration deployments.

This approach of networking particularly represents a new architecture that can also be related to the OSI reference model as shown in figure 2.11. The lowest layer is the *data plane* outlining the physical infrastructure. It contains the network devices and is responsible for transmitting the data packets between the infrastructure nodes. Network devices in the data plane do not make any routing or switching decisions. They execute actions received from the controller, including routing and switching orders. The data layer can be associated with the OSI physical layer.

The second layer in SDN is the *control plane*. It represents the main intelligence of the network. It pushes actions to the data plane and provides functionality to the management plane. The control plane includes the controller for the SDN environment, and exploits the SDN interfaces to exchange information with the network devices. Decisions that influence routing and switching are taken in this plane. It contains a set of common functions among network devices, and creates a general specification which can be implemented by a multitude of vendors. Table 2.1 lists a representative subset of available SDN actions for the OpenFlow 1.3 API [73]. It can be seen as a flow control

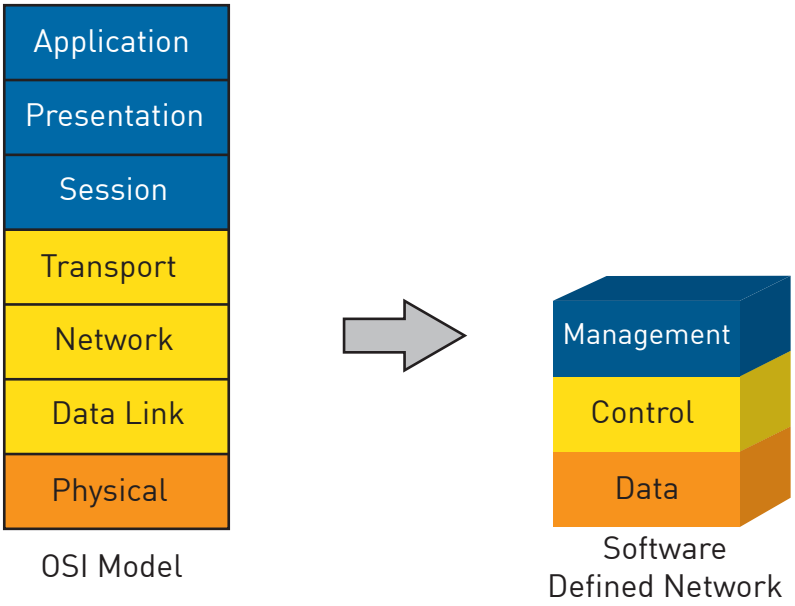

FIGURE 2.11: Correspondences between the ISO/OSI model and the SDN architecture.

entity and is affected by the management layer. The control plane can be associated with the OSI layer 2-4.

The *management layer* provides the interface for further use by humans or external applications. It is possible to create load balancing or anti-virus application in this layer that forward a set of instructions to the control plane, in order to implement their functionality. This separation forced in SDN environments leads to better network control as it provides an entity having global knowledge to efficiently maintain the network responsibilities.

The controller of an SDN environment is the sink of any event in the network and it always has a precise view on the network, including all links, nodes, and corresponding statistics. The controller can either be a separate hardware device or a software implementation running on a device connected to the control network. It manages all communication between any nodes in the network by influencing forwarding actions on the network nodes. The controller can instruct the nodes with a large set of actions specified by the API as presented in table 2.1. These actions are stored in the SDN nodes that set up the network infrastructure. Each node maintains a *flow table* that stores entries considering network streams passing this node. The stream can be identified by a composition of parameters covered by the OSI layers 2-4. This flow table associates the network stream with an action as presented in table 2.1. The node matches incoming packets against the flow table entries and, if the match was successfull, applies the corresponding action to the packet. The most prominent API for Software Defined

Flag	Description
OFPAT_OUTPUT	Output packet to switch port.
OFPAT_[PUSH | POP]_VLAN	Push a new VLAN tag / Pop the outer VLAN tag.
OFPAT_[PUSH | POP]_MPLS	Push a new MPLS tag / Pop the outer MPLS tag.
OFPAT_SET_NW_TTL	Set IP TTL value.
OFPAT_SET_FIELD	Set a header field using OXM TLV format.
OFPC_FLOW_STATS	Flow statistics.
OFPC_PORT_STATS	Flow statistics.
OFPC_TABLE_STATS	Flow statistics.

TABLE 2.1: Small subset of OpenFlow 1.3 supported Software Defined Network actions and statistics [22, 73].

Networks is *OpenFlow* [22]. The interface to the data plane includes the network device management and uses vendor-specific or open standardized interface structures.

Software Defined Networks provide a large set of flexible functionality to individually modify transmission behavior of network streams. Hence, it serves as an adequate basis to experiment with future transmission techniques as presented in this thesis. Section 7.2 introduces an SDN environment developed and used during this thesis and the project *OpenFlow@SaarlandUniversity*[7]. It also presents a set of different SDN-related applications and tools that have been developed in order to implement the proposed optimization strategies.

2.8 Network Functions Virtualization

Networks contain a large set of bare-metal devices from different vendors, including closed-source implementations and designs. Developing and deploying new services in such a traditional and static environment requires new hardware devices and software implementations. Network services can be expressed and defined as a *Network Function (NF)*. A network function implements a specific and well-known functionality in the network, such as firewall mechanisms, load balancing, DHCP [74], or DNS [4, 5]. Traditional networks consequently contain multiple network functions based on hardware devices, either in form of servers or pure network infrastructure, such as routers and switches.

As hardware life-time cycles are becoming shorter and the need for additional services is constantly growing, the *European Telecommunications Standards Institute (ETSI)* proposes in [24] the concept of *Network Function Virtualization (NFV)*. Currently used network functionality is connected, maintained, and used statically. It can neither react

[7] http://www.openflow.uni-saarland.de (2015/01/29)

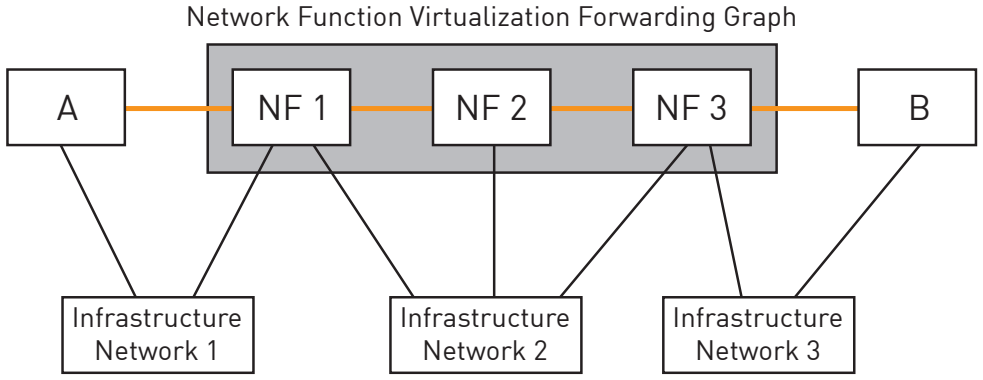

FIGURE 2.12: Network Function Virtualization Forwarding Graph (NFV-FG).

flexibly to varying network conditions, nor is it possible to seamlessly switch to another infrastructure. The virtualization of network functions implies the implementation of required functionality in software, that finally runs on top of a unified infrastructure. As a consequence *Virtualized Network Functions (VNF)*[8] enable the decoupling of network functionality into a software- and hardware-related part.

Following [24], *Network Function Virtualization* targets a fast innovation of new services, the standardization of open interfaces between *Virtualized Network Functions* and the infrastructure. Additionally, it is seen as an improvement considering financial objectives compared to hardware-based deployment of functionality.

When a network service consists of multiple functional blocks, each block is represented by an individual *Network Function*. These can be either arranged in series in order to force them to be gradually executed, or as an unordered set. In case they are arranged in series the structure is called *Network Function Virtualization Forwarding Graph (NFV-FG)* [24] as depicted in figure 2.12.

Individual *Network Functions (NF)* are connected via infrastructure networks. *Network Function Virtualization* is supposed to happen in the *network domain* and not in the *user domain*. Subsequently, NFs are present in network environments, but not in user's end devices [23].

The VNFs run inside a *Network Function Virtualization Infrastructure (NFVI)*-based on a virtualization layer. End devices use the NFs via network infrastructures and establish a logical and transparent link between end devices. This abstraction allows hiding the exact physical location of the VNF or device inside the environment without notifications to the connected end devices. The consequence is the usage of a broader set of physical resources and a higher spatial distribution at lower maintenance effort.

[8] Hint: Do not confuse with *Network Function Virtualization (NFV)* that describes the whole concept.

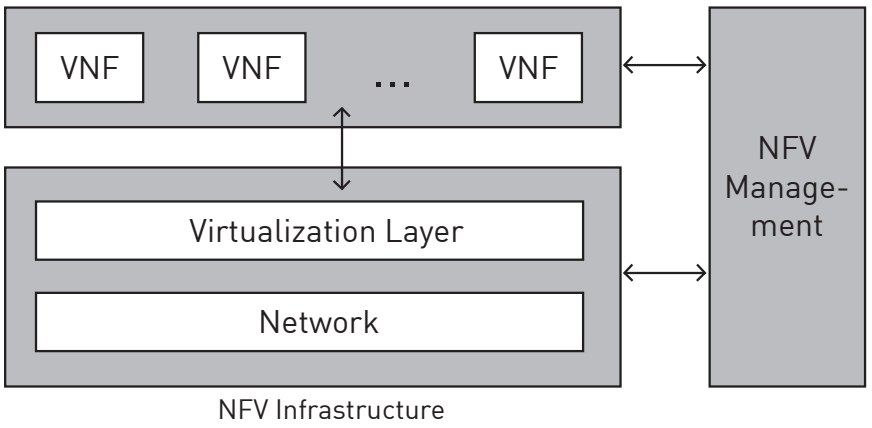

FIGURE 2.13: Network Function Virtualization Framework.

Figure 2.13 illustrates the schematic NFV framework with its components VNF, NFVI, and management.

The advantages of NFV are a shorter time to market for product developments, a near real-time network configuration on demand, and the ability to run products of different development stages on the same infrastructure [24]. NFV only scales in case all network functions can be virtualized and the network stability is important in order to provide a reliable infrastructure for all *Virtualized Network Functions*.

Chapter 3

Multimedia Transmissions

In the preceding chapter the underlying data network fundamentals have been discussed. These enable the reliable exchange of a large variety of information. This chapter focuses on a special type of data: *multimedia* information. The transmission of multimedia data requires a specialized handling, as this content type generally does not tolerate high latencies, large packet loss probabilities or jitter. Multimedia combines different streams of information. It contains access to dynamic content such as voice, video, or other data. Multimedia applications typically include *video conferencing*, *interactive TV*, *video on demand*, and different *electronic publications*.

It is generally supposed to combine *telephony*, *video*, and *data* services by the term multimedia as presented in figure 3.1. Depending on the application's type, multimedia content can have real-time character and specific interactivity requirements. This implies new challenges for network infrastructures that are primarily designed for text-based data exchange.

This chapter highlights basic multimedia application scenarios and their main transmission characteristics. It presents the most prominent transport protocols and an overview of the basic *Quality of Service (QoS)* principles, that are closely related with multimedia transmissions.

3.1 Multimedia Applications

Early network applications, such as *Email* [75, 76], *FTP* [77], or *Telnet* [78] enabled the transport of text-based information over network infrastructures. In the course of time, the primary information type has significantly switched to audio-visual data with a highly interactive character and strict requirements for the transmission environment.

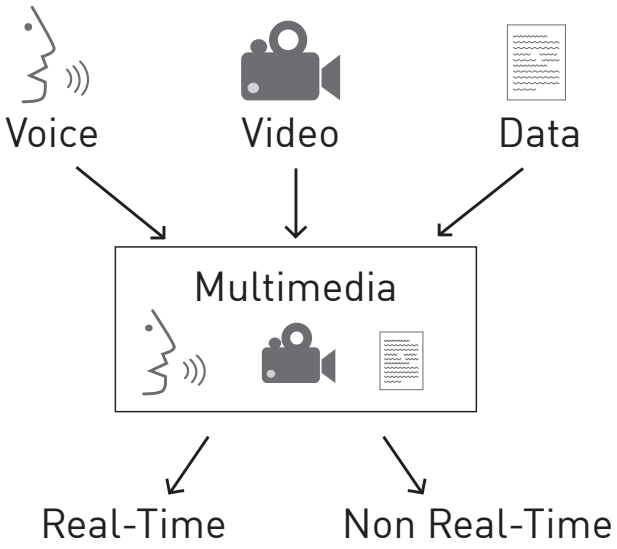

FIGURE 3.1: Overview of different types of multimedia applications.

According to recent forecasts [79], the amount of multimedia data carried within the global network traffic is continuously and remarkably increasing. As an example, in 2018 approximately one million minutes of video data will be present every second on the Internet [79]. *Multimedia applications* describe a fundamental class of applications that handle audio-visual information. Popular multimedia representatives are *voice*, *video-on-demand*, and *live-streaming* applications.

3.1.1 Voice over IP

In the beginnings of the Internet, data was mainly sent over *Public Switched Telephone Networks (PSTN)*, e.g. based on the ITU-T recommendation V.90 [80]. This scenario has been changed in a way that the currently predominant number of telephone connections are established via packet-switched networks, generally based on the *Internet Protocol (IP)* [1]. Voice data sent via IP-based networks is called *Voice over IP (VoIP)* [81].

The most significant difference between data and telephone networks is their nature of establishing connectivity between the source and sink. IP networks are *packet-based*. They insert information into variable-sized packets and use statistical multiplexing to transmit these packets towards the sink. The available data rate consequently depends on the current activity on the links and can vary over time.

PSTNs in contrast are *circuit-switched* networks. They completely occupy the network connection as long as the current call is active. Once established, this in fact enables

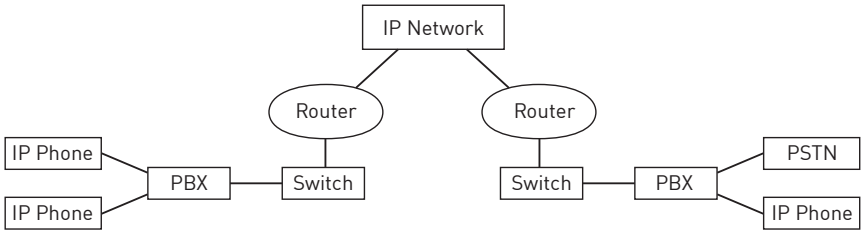

FIGURE 3.2: Voice over IP (VoIP) scheme.

a reliable and fast voice connection between both communication partners without impairments. As an occupied line cannot be used for another call, this scheme is highly inefficient and costly. A sufficient number of available lines must be always kept in stock in order to avoid call rejects.

Voice over IP (VoIP) is a technology that uses digital signal processors to segment the audio information and store them in variable-sized packets. Figure 3.2 schematically illustrates the VoIP transmission process. VoIP signaling is typically handled with H.323 [82] and the *Session Initiation Protocol (SIP)* [83], whereas the transmission is based on *Real-Time Transport Protocol (RTP)* [28], which relies on the *User Datagram Protocol (UDP)* [2]. These protocols are discussed in section 3.4. A prominent representative application for VoIP is *Skype*[1].

IP networks can generally also be substituted by *Asynchronous Transfer Mode (ATM)* [40] or *Frame Relay (FR)* [84] networks. In contrast to IP networks, these network types are predominately used in provider backbone networks and are rarely available to end users. According to ITU-T recommendation G.114 [85] the one-way delay limit of a transmission carrying voice information should be at most 150 *ms*. As *jitter* is a critical transmission parameter, multiple service providers fix an upper limit in the order of 0.5 *ms* - 2 *ms* [86]. *Packet loss* is also crucial for VoIP communication. Following a suggestion from Cisco Systems[2], the default ITU-T G.729 codec [87] requires a loss probability of significantly less than 1% to avoid any audible errors [88].

VoIP eases the communication between different partners. It harnesses the already available IP infrastructure for routing between peers and enables a high interconnection of devices and technologies from different vendors. VoIP is strictly standardized, what consequently implies a reliable vendor-independent large scale deployment. *Voice over Frame Relay (VoFR)* [89] and *Voice over ATM (VoATM)* [90] lack such a stringent specification.

[1] http://download.skype.com/share/business/guides/skype-connect-technical-datasheet.pdf (2015/01/29)

[2] http://www.cisco.com (2015/01/29)

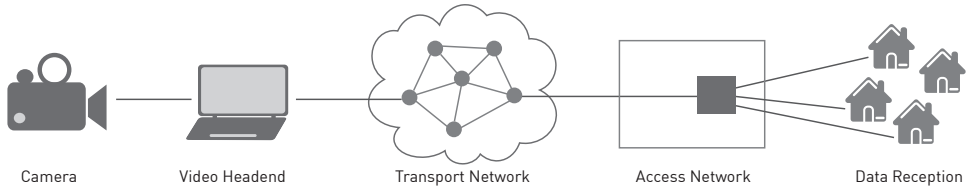

FIGURE 3.3: Sample IP TV scheme including different domains.

3.1.2 Internet Protocol Television

Internet Protocol Television (IPTV) describes the transport of television content via packet-switched networks like the Internet. As the name implies, it uses IP network infrastructures. IPTV arose as the answer to the need of having video content at any place, at any time, and on any device. As Internet access is approximately available at nearly every location in the world, the solution seems to be straightforward.

As IPTV is an IP-based broadband Internet approach, it uses digitally switched infrastructures that significantly differs from the traditional channel based transmission environments for legacy broadcasting television. *Set Top Boxes (STB)* decode IP video packets and supersede the need for a tuning process. IPTV is highly interactive and can be flexibly personalized to individual customer requirements. It also efficiently works with respect to network resources, as content is solely transmitted to users that have sent a request.

A typical IPTV infrastructure consists of five different partitions as depicted in figure 3.3. These five domains are:

1. **Data Source:** This part generates the video material or forks it from another transmission medium, such as satellites and ensures that the initial quality is sufficient.

2. **Video Headend:** The video headend compresses and probably encrypts the raw material from the data source. It additionally encodes the material in order to send it over IP networks. In case the material is not streamed live, it can also be stored for future access.

3. **Transport Network:** This is the core IP backbone network of the service provider. Adequate *Quality of Service (QoS)* regulations are required in order to minimize packet loss and jitter as well as providing sufficient bandwidth and data rate for the transmission.

4. **Access Network:** The access network represents the so called *last mile* before reaching the end user. Data is potentially delivered by a multitude of different

access technologies such as *xDSL* [91, 92, 93] or *FTTx* [94]. This requires adequate protocol support for connectivity.

5. **Data Reception:** As the user receives the data, it is generally distributed on-premises to multiple end devices, such as Set Top Boxes (STB), mobile phones or desktop computers.

This structure includes three networks, each with individual characteristics. The transport and access networks are typically error free, but introduce a non-neglectable delay. The in-house data infrastructure potentially generates a small latency while causing significant packet losses, especially in case of wireless LANs. The bidirectional transmission character of IP networks enables the application of flexible error correction schemes, such as *Automatic Repeat Requests (ARQ)* [34], and is not limited to pure unidirectional schemes as *Forward Error Correction (FEC)* [54].

Internet Protocol TV must be dissociated from WebTV. In general, IPTV incorporates a specific Quality of Service minimum and is managed by an Internet Service Provider (ISP). WebTV is the transmission of freely available content over the Internet following the best-effort principle without any quality guarantees.

3.2 Transmission Characteristics

Each application type includes different aspects that influence the characteristics and requirements of a network transmission. In the following, the physical characteristics and associated challenges for multimedia transmissions are presented. Additionally, an overview containing the most prominent streaming approaches is given. This section extends the initial discussion of network characteristics in section 2.3.

3.2.1 Properties

Transmitting data means to move information between at least two distant locations. A significant property of each transmission is latency. It is constituted by the physical propagation delay, queuing delays, and the size of the used information chunks. Figure 3.4 illustrates different ingredients of the latency. The challenge for most transmissions is to significantly reduce the latency. Queues and other side parameters of the transmission are hard to modify. For example, a suitable method to modify the physical latency is to replace the physical transmission medium, e.g. OSI layer 1, in order to improve its capacity and data rate properties.

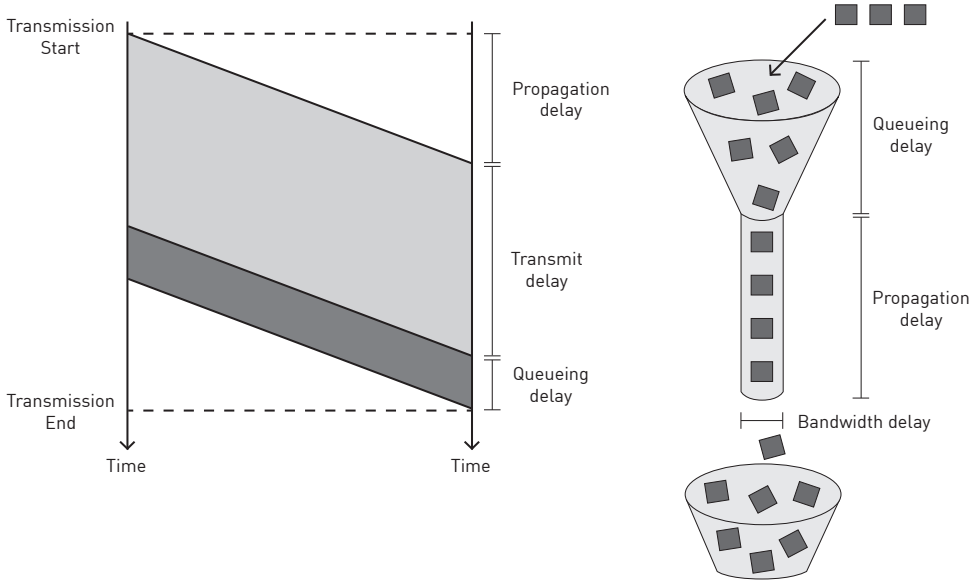

FIGURE 3.4: Ingredients of the transmission latency.

Information moving through networks can also be lost or impaired. This may happen due to transmission interruptions, e.g. caused by bit flips in packets, or overloaded network segments or devices. In this case, information is dropped at these devices, because the available capacities are not sufficient anymore.

Error correction schemes try to cope with occurring packet losses due to physical factors, but can also be applied to congested networks. In this case, an ARQ scheme, as discussed in section 2.5, retransmits dropped packets. Constant congestion-initiated packet losses can only be prevented by extending the network infrastructure with larger capacity resources.

Packet-switched networks potentially select an individual network route for each packet. Some routes may include more hops or links than others. Hence, it is natural that packets taking individual routes arrive out of order at the receiver.

Another serious issue for multimedia transmissions is packet jitter, i.e. the time variation between the receptions of two original subsequent packets. In case of latency-sensitive transmissions, packet jitter highly influences the playback quality. Large jitter implies a stalling content playback.

3.2.2 Challenges

Multimedia has special requirements towards the underlying network architecture. In case of a voice transmission, it unveils that this content type is highly sensitive to latency.

An adequate voice-based communication includes a strict and careful reassembling of tones and pauses in order to avoid unnatural speech characteristics. Large packet latencies, highly varying jitter, or packet losses cause an insufficient quality of the voice communication.

By default, IP networks are designed to transport data bursts, such as text documents or file downloads, thus being not highly sufficient to carry a steady stream of voice or video information. A trivial approach to overcome these limitations is to increase the available bandwidth and data rate at critical locations within the network. This is a possible method to cope with large latencies and data congestion situations in transport backbone networks. However, this is impractical in access and home networks. The implementation of *evolutionary* and *revolutionary* optimization approaches is considered to be valuable. An appropriate handling of these challenges lead to a higher transmission efficiency, compared to a trivial extension of transport capacities.

New network technologies and transport protocols as discussed in sections 2.7 and 3.5 are a first approach to evolutionarily modify existing network environments. Chapter 4 and 5 present approaches for multimedia transmission scenarios in the future media Internet.

3.3 Transmission Scenarios

A multimedia transmission can be handled via different approaches, depending on its requirements. A basic mechanism is to store a previously captured video on a server and offer the content as a file download. In case the transmission contains real-time video information, this procedure is not sufficient, and more complex techniques have to be involved in order to reliably distribute the data to the receivers.

With *on-demand streaming*, previously stored video material is requested from a server or from a cloud application, and delivered via a network to the end user. The customer is able to control the streaming transmission up to a certain extent. This implies to *pause* or *stop* the transmission, as well as perform a *fast-forward* or *reverse search* within the stream. In either case, individual commands are sent to the server and it selects the appropriate data chunks requested by the receiver. The server application subsequently delivers the required portion of the video directly to the user. In this case, the video material is always cached on the receiver side and retrieved from this cache before displaying it. As a special feature, with on-demand streaming the user is able to start watching the content when sufficient data has been loaded from the server. There is no need to download the whole file first. This leads to a significant

latency reduction. If the cache runs out of data, the video stalls until upcoming data has filled the cache again. This involves a sophisticated application and buffer design. On-demand streaming is generally implemented by a unicast distribution scheme based on controlling features. In case multiple users demand the same video content, an individual unicast transmission is typically established to each of them. The probably most prominent protocol used with on-demand streaming is the *Hyper Text Transport Protocol (HTTP)* [29]. An exemplary application is defined in ISO/IEC23009-1 [95]. It is called MPEG-DASH [27] and represents a standard for adaptive bit rate streaming for high quality content.

In contrast to on-demand streaming, *live streaming* includes the delivery of data in near real-time. It is less interactive as it does not support fast-forwarding or replaying live content. Live streaming is often implemented as a multicast service in closed networks in order to lower the required data rate. It often uses protocols that allow a fast packet delivery, such as UDP [2]. There are additional protocols that are developed for real-time transmission and control of data, such as *RTP*, *RTSP*, *RTCP*, and *SIP*. These protocols are discussed in section 3.4.

3.4 Multimedia Transport Protocols

Multimedia transport protocols rely on the functionality provided by the network layer. In packet-switched networks the *Internet Protocol (IP)* [1] is usually implemented. The protocols transport data between devices. Some protocols additionally provide functionality to handle congestion and flow control, packet losses, and distortions, as well as packet duplication and reordering. They provide fundamental mechanisms to send data with specific requirements for reliability or delay.

The specification of a general-purpose multimedia transport protocol is a challenging task, as it must consider a broad range of different applications and individual requirements. Predefining a fixed subset of parameters, e.g. for encoding and compression, heavily restricts the application's flexibility to adjust the sending behavior to different environmental parameters. It is more convenient to use a negotiation procedure to enable an information exchange between sender and receiver. Multimedia information exchange considers multiple aspects.

Section 4.5 presents a generic optimization approach called *Network Supported Congestion Avoidance (NSCA)*, that enables a precise negotiation between applications and network environments. It implies a multimedia transmission adhering to the network infrastructure and its current status, as well as the application's requirements.

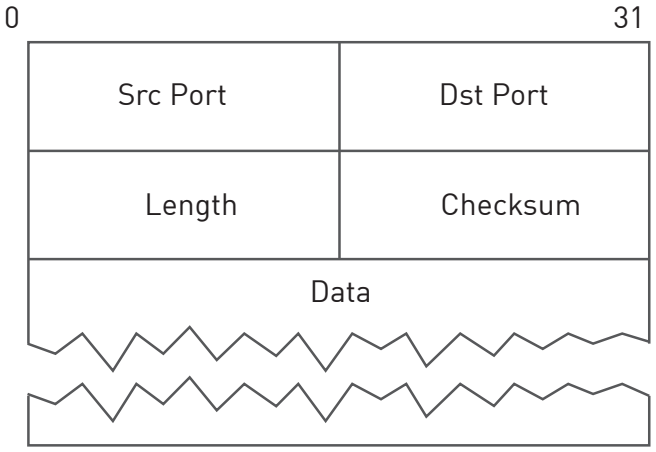

FIGURE 3.5: The UDP packet header.

The received data must be related to a specific timing characteristic and ordering to enable a correct and adequate playback. When packet losses occur during transmission, the receiving device must be able to detect these losses and start an appropriate loss-handling procedure. In case application specific data chunks are longer than the common network packet size, e.g. the maximum transfer unit (MTU), it is inevitable to split them into multiple packets. The protocol must be able to indicate these frame boundaries. One missing packet of a sequence, that contains frame data, can generally lead to a distorted playback frame.

Besides all of these aspects it is unavoidable to implement all these features in a way that the available network bandwidth is used efficiently. There is a large set of different specialized multimedia transport protocols. *PRRT* [66] is a transport protocol specially developed for the purpose of transmitting real-time multimedia content. PRRT contains a sophisticated adaptive hybrid error correction, congestion control, and a dynamic parameter calculation of an optimal network-aware transport of multimedia content. PRRT is presented in section 3.5.

In the following a subset of the most prominent default protocols for video and audio transmission, including their control protocols, are presented. Further protocols can be found in [96, 97, 98, 99].

3.4.1 User Datagram Protocol

The *User Datagram Protocol (UDP)* represents a basic transport protocol. It purely relies on the best-effort mechanisms of the underlying network and does not feature any sophisticated functionality. It establishes a host-to-host communication link and was initially specified in RFC 768 [2]. It does neither contain any error or congestion control,

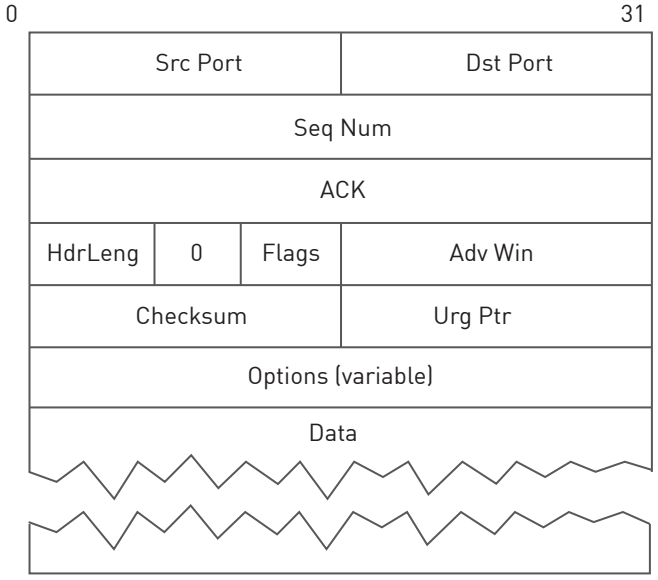

FIGURE 3.6: The TCP packet header.

nor flow management functionality. Ordered packet delivery is also not targeted by this protocol. It transports data from one device to another, while leaving more complex mechanisms to the remaining OSI layers as discussed in section 2.1. Figure 3.5 illustrates the UDP packet header and its corresponding simplicity.

Despite being a general purpose transport protocol, UDP remains a good choice for multimedia applications. Owing to its simple structure, it delivers packets with low latency. On the other hand, the application must be able to tolerate packet losses as UDP does not provide any functionality for a reliable transmission. As multimedia applications assume low latency to be more important than full reliability, UDP is one of the most prominent transport protocols for audio and visual information exchange. Multimedia applications typically use the *Real-Time Transport Protocol (RTP)* that runs on top of UDP and adds more functionality. RTP is discussed in section 3.4.3.

3.4.2 Transport Control Protocol

The *Transport Control Protocol (TCP)* is a more sophisticated protocol than UDP and has been specified in RFC 793 and RFC 1323 [3, 100]. It releases the application from handling situations with missing or disordered packets by providing functionality for a reliable and in-order delivery of packets. It offers mechanisms for flow and congestion control, as well as multiplexing different applications while connecting two hosts. Due to the flow control, TCP allows the receiver to restrict the sending rate of the sender in order to protect local buffers from overflowing. The congestion control manages the sending rate according to the underlying network conditions. It avoids dropped packets

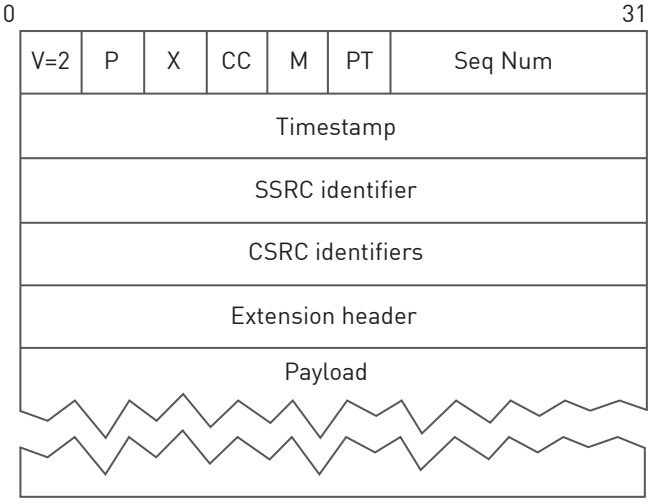

FIGURE 3.7: The RTP packet header.

owing to overloaded network devices as switches and routers by an early sending data rate reduction. TCP is probably the most widely used transport protocol in current IP networks. It is used with mailing [101, 102], network services [4, 103, 83, 104, 15] and nearly all world wide web applications [29, 105, 106]. The protocol has a full-duplex characteristic, that is data can be sent bidirectionally between the end hosts using a single connection. Figure 3.6 illustrates the TCP header. It shows a more complex structure than UDP due to the required fields for controlling mechanisms. As streaming multimedia data comes down to a stream-oriented transmission, TCP immediately suggests itself to be used in this context. This protocol provides reliability that is directly linked with potentially large delays due to retransmission cycles. As multimedia transmissions prefer skipping frames over waiting for packets to be retransmitted, TCP is not an adequate choice for this type of content, especially in case of real-time constraints.

3.4.3 Real-Time Transport Protocol

The *Real-Time Transport Protocol (RTP)* is defined in RFC 3550 [28]. It is a widely deployed protocol and contains specialized end-to-end functionality for multimedia applications. It generally runs on top of UDP and uses the already available application independent network functionality provided by this transport protocol. It is responsible for the transport of the streaming data and works in conjunction with the *Real-Time Control Protocol (RTCP)*. RTCP allows the frequent exchange of statistical information between sender and receiver. RTP has been designed to support a wide range of multimedia applications. Its specification supports flexible mechanisms to ease the development of new multimedia applications, and avoids modifications at hard-to-reach locations inside the protocol stack. The RTP packet header is depicted in figure 3.7. The

Real-Time Transport Protocol also describes *profiles*. A prominent profile representative is *RTP Audio Video Profile (RTP/AVP)* as described in RFC3551 [107].

RTP represents the successful adaptation of *Application Level Framing (ALF)*. *ALF* is an architectural principle that allows the applications to use its individual semantics for the utilization of network protocols. It unveils that the use of general-purpose transport protocols should be avoided in order to achieve efficient multimedia applications.

3.4.4 Real-Time Control and Streaming Protocol

The *Real-Time Control Protocol (RTCP)*, as defined in RFC 3550 [28], augments a multimedia data stream. This protocol includes basic identification and control functionality as well as a reporting scheme to monitor a connection. RTCP is designed to work independently of the underlying layer, but it generally relies on UDP. The *Real-Time Control Protocol* has multiple different packet types [28].

The main types are:

- **Sender Reports (SR):** It is sent in case a device has sent data packets in the interval starting from the last report or in case any packets have been sent.
- **Receiver Reports (RR):** This report resembles the sender report except for the sender information section that is used for active sender. The receiver report is sent in case no sender report has been sent.
- **Source Description (SDES):** This packet carries sender information, e.g. *Canonical End-Point Identifier (CNAME)* or *User Name (NAME)*
- **Goodbye (BYE):** This packet type indicates that one or more sources are no longer active.
- **Application Specific Control Packets (APP):** This experimental packet type contains application specific information to develop new features without the need to register new packet types.

Derived from the main packet type overview, RTCP offers additional functionalities for multimedia streams. *Feedback* can be sent between sending and receiving applications in the network. It enables a Quality of Service and performance monitoring of an RTP session. *Synchronization* is offered between different media streams that have been originated by the same server. In this case, RTCP extends the RTP `SourceID` field to support the synchronization of different streams with potentially different clocks.

The *Real-Time Streaming Protocol (RTSP)* is defined in RFC 2326 [108]. It establishes and controls data streams with real-time requirements. Applications can use RTSP to remotely control a stream over the network. It uses either connection-less or connection-oriented transport protocols. RTSP controlled streams are generally sent via RTP, but any other transport protocol can be used to stream the data. RTSP is similar to HTTP and it analogously employs a request/response mechanism.

RTSP provides the following functionalities:

- **Retrieval of Media:** Clients are able to request information or streams from a server. The server then provides all required information, such as network addresses.
- **Invitations to Conferences:** Clients can be invited to join a particular presentation. This mode can be used to create online teaching applications.
- **Addition of Media to Presentations:** In case of live presentations the server can announce additional services according to the ongoing presentation via RTSP to the client.

The *Real-Time Streaming Protocol* defines a large set of different methods to request a method or operation [108]. For example, the client (C) can ask the server (S) about its remote control capabilities:

```
C->S:  OPTIONS * RTSP/1.0
       CSeq: 1
       Require: implicit-play
       Proxy-Require: gzipped-messages

S->C:  RTSP/1.0 200 OK
       CSeq: 1
       Public: DESCRIBE, SETUP, TEARDOWN, PLAY, PAUSE
```

Subsequently, the server tells the client that it can handle `DESCRIBE, SETUP, TEARDOWN, PLAY,` and `PAUSE` methods.

Starting a live streaming via RTSP includes the following messages:

```
C->S:  PLAY rtsp://audio.example.com/twister.en RTSP/1.0
       CSeq: 833
       Session: 12345678
       Range: smpte=0:10:20-;time=19970123T153600Z

S->C:  RTSP/1.0 200 OK
       CSeq: 833
       Date: 23 Jan 1997 15:35:06 GMT
       Range: smpte=0:10:22-;time=19970123T153600Z
```

RTSP contains a large set of different commands and actions. More specifications are presented in [108].

3.4.5 Session Initiation Protocol

The *Session Initiation Protocol (SIP)* is an application-layer control protocol and specified by the IETF in RFC 3261 [83]. It works with both IPv4 and IPv6 and is primarily intended for session management. SIP enables user agents to discover peers in the network and to ease contacting them. It works completely autonomous from the underlying transport layer and the corresponding session that is to be established by SIP. It flexibly offers services, such as establishment, modification, and termination of sessions.

The main concept of SIP is to constitute a general-purpose tool for a wide variety of applications requiring session management. SIP does not provide any individual services, but offers a set of primitives to applications that can be used to create customized services. SIP works in a similar manner as HTTP [29] and uses request and response messages to exchange information. A significant difference is, that HTTP has been designed for *machine-machine* communication, whereas SIP focuses on *human-human* communication. It is not sufficient to solely detect the target machine, as the designated user must also be identified. The functionality of SIP can be coarsely subdivided into two parts: *user management* and *session management*. SIP provides functionality to answer the following most prominent questions within session management:

- **User Location:** How can the user be reached to establish an individual connection and which device is used by the user?
- **User Availability:** Is the demanded user ready and able to enter the session?

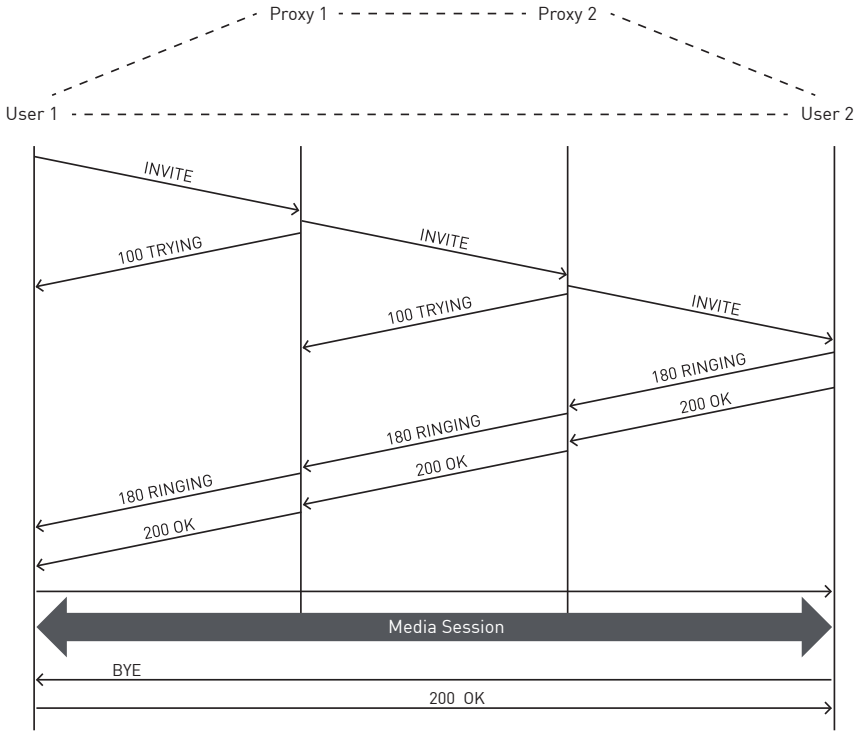

FIGURE 3.8: An exemplary SIP message flow.

- **User Capabilities:** Which media types are supported by the user and its device? Which codecs can be used for the video and audio transmission?
- **Session Setup:** How can the session parameters be negotiated? Which transport ports should be used?
- **Session Management:** How can a session be modified? How to implement a call forwarding or parameter change?

Figure 3.8 illustrates a high level SIP message flow between two user agents.

A typical SIP `INVITE` message [83] looks as follows:

```
INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bK776asdhds
Max-Forwards: 70
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710@pc33.atlanta.com
CSeq: 314159 INVITE
Contact: <sip:alice@pc33.atlanta.com>
Content-Type: application/sdp
Content-Length: 142
```

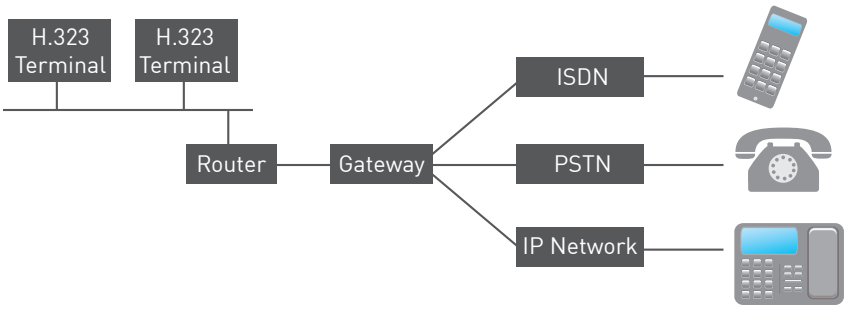

FIGURE 3.9: An exemplary H.323 network environment.

A typical SIP `OK` message [83] looks as follows:

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP server10.biloxi.com
 ;branch=z9hG4bKnashds8;received=192.0.2.3
Via: SIP/2.0/UDP bigbox3.site3.atlanta.com
 ;branch=z9hG4bK77ef4c2312983.1;received=192.0.2.2
Via: SIP/2.0/UDP pc33.atlanta.com
 ;branch=z9hG4bK776asdhds ;received=192.0.2.1
To: Bob <sip:bob@biloxi.com>;tag=a6c85cf
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710@pc33.atlanta.com
CSeq: 314159 INVITE
Contact: <sip:bob@192.0.2.4>
Content-Type: application/sdp
Content-Length: 131
```

SIP packets contain plain text messages including all relevant information for the following session. It is principally possible to parse this information from SIP messages in order to pre-optimize the network according to the subsequent data transmission.

3.4.6 H.323

The H.323 protocol describes the ITU recommendation for multimedia-based communications over packet-switched networks [82]. It draws on multiple other ITU recommendations such as H.225 [109] and H.245 [110]. Owing to this high interdependence, it appears to be a complex protocol. H.323 is adapted to support IP telephony over the Internet. It is also the first protocol for this purpose, that uses RTP to transmit the data over the network.

H.323 describes three different types of devices:

- **Terminals:** These are the end devices of an H.323 communication. Terminals can directly communicate with each other in a peer-to-peer fashion. They also frequently contact a *Gatekeeper*.
- **Gatekeepers:** These devices mediate the communications in the network between the terminals. They translate address formats, control calls and their bandwidth requirements as well as help terminals to find a *gateway*.
- **Gateways:** A gateway connects an H.323 network to another network. The most prominent network interconnection is from H.323 to a *Public Switched Telephone Network (PSTN)* to establish the link between IP telephony and the conventional phone devices in a PSTN.

Figure 3.9 schematically illustrates the interaction of H.323 devices in a network.

3.5 Predictably Reliable Real-time Transport

With today's transport protocols the transmission of real-time multimedia streams over IP networks is a challenging task. Their architecture primarily focuses on strict reliability and does not tolerate any residual packet loss. But this loss tolerance is one of the main characteristics of multimedia applications. This specification leads to a rigid error correction mechanism that implies unpredictable delivery latency in case data is being sent over lossy network segments. As each multimedia application has individual requirements for transmission latency and residual loss, mechanisms are needed to provide these information to the transport layer. Today's protocols adhere to the ISO/OSI layer model as discussed in section 2.1, but this model prevents cross-layer communication. Applications are not able to announce their requirements about delivery latency and packet loss to the network.

Predictably Reliable Real-time Transport (PRRT) is a principle that has been developed at the Telecommunications Lab[3] to overcome these limitations of today's transport protocols. PRRT also includes a transport protocol that implements the corresponding principle. Figure 3.10 illustrates the core framework of PRRT. It is an innovative, capacity-approaching transmission scheme, that is able to implement an application specific level of reliability, while providing a strict upper limit for delivery delay. An *Adaptive Hybrid Error Correction (AHEC) framework* has been investigated in [36]. It

[3] http://www.nt.uni-saarland.de (2015/01/29)

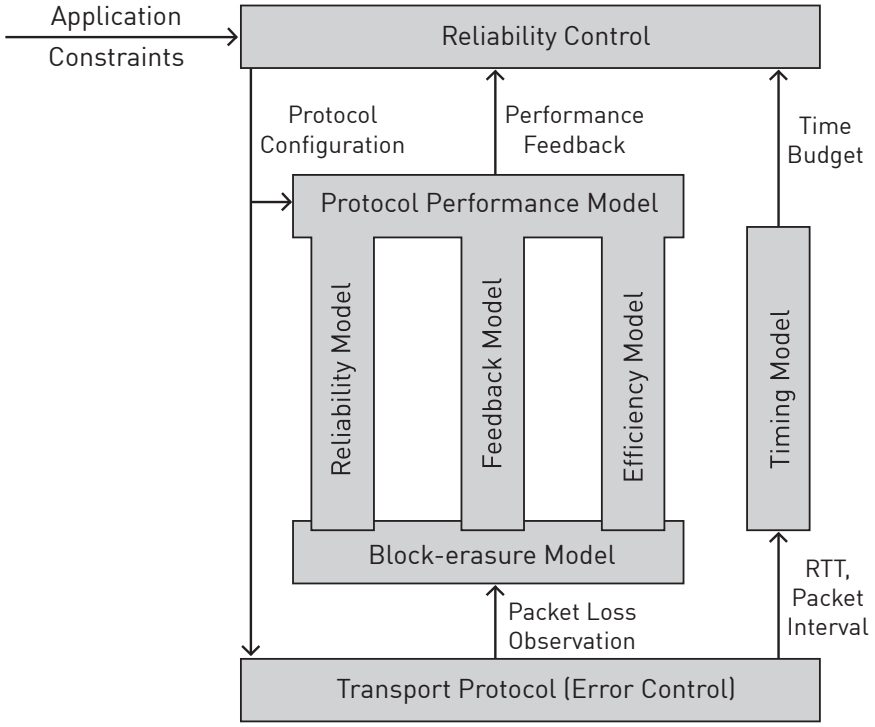

FIGURE 3.10: The PRRT core framework following [66].

constitutes the theoretical foundation of the PRRT principle. This framework combines both, proactive and reactive error correction approaches. Both approaches are discussed in section 2.5. PRRT works with one-to-one as well as one-to-many communication channels.

The focus of PRRT is on predictable reliability for multimedia transmissions while considering strict application individual delay constraints. As PRRT uses hybrid error coding, a set of parameters is required for both *Forward Error Correction (FEC)* [54] and *Automatic Repeat Request (ARQ)* [34], according to the given application requirements. The protocol must efficiently use network resources, which consequently implies the application of an optimized set of parameters. This optimization is based on a stochastic process that reflects the protocol behavior in case of packet loss. The stochastic process is represented by a block erasure model [66], that adequately follows real world environments. This model is also used to calculate the achieved residual packet loss rate and the corresponding coding overhead. The obtained set of parameters is frequently evaluated by a reliability control policy. This ensures a valid and productive parameter selection for the present network conditions and application requirements. The protocol parameters are found as a solution of a combinatorial optimization problem. The search algorithm has explicit knowledge of the search space. It is applied to find appropriate solutions for the given optimization problem.

These results were turned into a media-oriented transport protocol stack as shown in [66]. The implementation can be freely downloaded from the Telecommunications Lab website[4].

In chapter 5 a network separation scheme is discussed that refers to the basic results of the AHEC framework.

3.6 Quality of Service Principles

Multimedia streaming applications produce a composition of audible and visual information along with control data. After the application has digitized the individual inputs, it eventually starts the generation of a continuous stream of data packets. This stream initially looks as any other ordinary data stream, but as it contains video and audio information, it demands high transmission rates. An implicit requirement for multimedia content is a timely delivery, a low packet loss rate, and a minimum of packet jitter. The network infrastructure must be able to appropriately deliver the multimedia data. Here, *appropriately* can have different meanings. Some applications do not tolerate any loss, whereas others can handle it up to a certain limit. Some applications allow several hundreds of milliseconds delay, whereas others assume a specific maximum transmission latency. Each definition directly influences the end user's experience. The transmission of multimedia contains multiple aspects such as queuing, spatial distribution, and error correction. All of them affect the *Quality of Service (QoS)* [40] since a complex of multiple independent characteristics must be considered.

A core implication of QoS is that the appropriate delivery of data on time significantly relies on the network infrastructure, including all components, e.g. switches, routers, and transport protocols. *Quality of Service* is additionally influenced by the end devices and their implementations.

Traditional IP networks consist of a decentralized switching and routing infrastructure and deliver packets following the best-effort principle. Each packet is considered individually. Thus, two consecutive datagrams of the same traffic stream may take different routes through the network. Both experience individual effects as delay, jitter, and loss. This does not represent an adequate transmission environment for multimedia data with strict transport requirements. The infrastructure must be extended by mechanisms to assure these multimedia *Quality of Service* requirements. *Quality of Service* approaches

[4] http://www.nt.uni-saarland.de/en/projects/running-projects/prrt.html (2015/01/29)

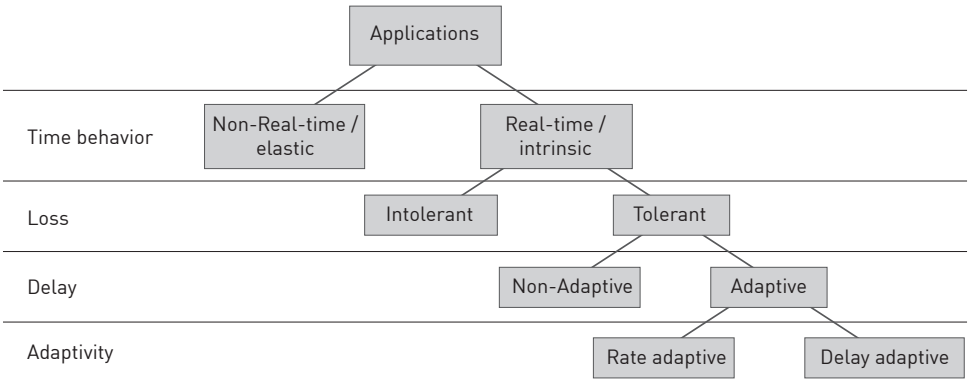

FIGURE 3.11: A Quality of Service (QoS) classification scheme following [64].

always contain an evolutionary characteristic as only applications that require prioritized transport parameter utilize QoS mechanisms. Other applications without QoS requirements use default approaches to transmit data to the destination.

Default applications can be categorized as shown in figure 3.11. They can be grouped according to their time characteristics as *elastic* or *intrinsic*. The former means the data must arrive correctly without strictly focusing on transmission latency. The latter assumes an intrinsic time behavior that predicts the transmission time. Downloading a file belongs to the *elastic* and streaming a video to *intrinsic* characteristics category.

In case there is a real-time requirement, applications can further be split into *loss tolerant* or *loss intolerant*. *Loss tolerant* implies that the application can handle losses up to a certain extent or it flags an error and uses mechanisms to recover this error.

This application type is further subdivided into *delay adaptive* or *non delay adaptive*. This implies, that it is possible to adjust the receiving and delivery behavior at the receiver to varying packet arrival times. This can be implemented by a sophisticated buffering policy.

Delay adaptive applications can be categorized into *rate adaptive* and *delay adaptive*. Rate adaptive implies the ability to adjust the coding rate of the multimedia data according to the present network conditions and thus lower the quality and required data rate in case the network is highly occupied. In case the application is delay adaptive, it is possible to buffer data until a satisfying playback is possible. This leads to an extension of the established best-effort principle on the present Internet and other wide area networks. Multiple different classes are inserted to the service model in order to support a multitude of individual applications. There are two main characteristics:

- **Fine-grained:** This class includes *Quality of Service* specifications for individual applications or flows.
- **Coarse-grained:** This class contains *Quality of Service* specifications for a wider and not precisely defined class of data, e.g. aggregated traffic.

In the course of time, these two basic approaches have been further developed. *Quality of Service* approaches focusing on fine-grained and coarse-grained are known as *Integrated Services (IntServ)* [11] and *Differentiated Services (DiffServ)* [111], respectively.

3.6.1 Integrated Services

This mechanism to prioritize data packets in a network has been developed by the IETF between 1994 and 1997 and is specified in RFC 1633 [11]. *Integrated Services (IntServ)* contain the specification of service classes for individual traffic flows and applications. The *IntServ* mechanism identifies two basic service classes:

- **Guaranteed Services:** This class contains directives for delay intolerant applications. The network ensures a maximum latency for the data packets and that no packet is dropped in a network device queue. It is defined in RFC 2212 [7].
- **Controlled Services:** This class includes specifications for delay tolerant and adaptive applications. It is assumed that the corresponding data traffic can be transmitted with no significant impact over slightly loaded network infrastructures. In this case, mechanisms such as *Weighted Fair Queuing (WFQ)* [64] are used to isolate the categorized traffic from the default traffic. An admission control scheme limits the number of streams in this class. It is defined in RFC 2211 [112].

Both approaches include the requirement to inform the network about the type of data the application is going to send. The network decides up to which level it is able to offer the required services. *Integrated Services* use the *Resource Reservation Protocol (RSVP)* [113] to establish a resource reservation within the network.

3.6.2 Differentiated Services

The *Differentiated Services (DiffServ)* approach assumes a small number of different traffic categories. In fact, it works with only two categories and each has individual QoS requirements and impacts on the routing and switching behavior.

Class	IPTD	IPDV	IPLR	IPER	IPRR	Application Scenario
QoS class 0	100 ms	50 ms	1×10^{-3}	1×10^{-4}	-	Real-time (VoIP)
QoS class 1	400 ms	50 ms	1×10^{-3}	1×10^{-4}	-	Real-time (VoIP)
QoS class 2	100 ms	-	1×10^{-3}	1×10^{-4}	-	Highly interactive
QoS class 3	400 ms	-	1×10^{-3}	1×10^{-4}	-	Interactive
QoS class 4	1 s	-	1×10^{-3}	1×10^{-4}	-	Low loss only
QoS class 5	-	-	-	-	-	Best-effort
QoS class 6	100 ms	50 ms	1×10^{-5}	1×10^{-6}	1×10^{-6}	IPTV
QoS class 7	400 ms	50 ms	1×10^{-5}	1×10^{-6}	1×10^{-6}	IPTV

TABLE 3.1: Quality of Service requirements for different classes according to ITU-T Y.1541 [121].

DiffServ is tailored to carry traffic with low-latency requirements in networks with parallel traditional best-effort traffic. Each packet carries the information about its affiliation in a header tag and identifies itself to routers when arriving at the device. The *DiffServ* information is set on administrative boundaries, such as inter-AS [114] routers. It is based on policies granting individual processing to specific packets. This special treatment for the packets is performed by the routers and summarized in the *Per-Hop Behaviors (PHB)* descriptions as defined in RFC 2475 [10]. PHB are also used in *Multi-Protocol Label Switching (MPLS)* networks [18, 115]. In production networks four main PHB schemes are used:

- **Default:** This PHB implies a traditional best-effort service for the packet. It is assigned to any packet that has no valid or known PHB header information.
- **Expedited Forwarding (EF):** This PHB has been specified in [116]. It defines a forwarding with a minimum of latency, packet loss, and jitter. This behavior is ensured by using rate limiting and priority queuing.
- **Voice Admit (VA):** In [117] the IETF specified the voice admit PHB with the same properties and routines as with *expedited forwarding* PHB, but with an extension to use a caller admission control mechanism.
- **Assured Forwarding (AF):** Initially defined in [118] and extended in [119] the AF PHB can be seen as a derivative of *Random Early Detection (RED)* [120]. Packet drop probabilities are used inside the queues of network devices to efficiently balance the traffic.

3.6.3 Service Classes

In ITU Y.1541 [121] the International Telecommunication Union (ITU) has recommended a set of *QoS* classes as presented in table 3.1. These service classes provide a set of criteria to be met by IP transmissions focusing on *transfer delay (IPTD)*, *delay variation (IPDV)*, *loss ratio (IPLR)*, *error ratio (IPER)*, and *reordering ratio (IPRR)*. As explained in section 2.3.2, IPER represents the ratio of erroneous, but received packets per total sent packets, whereas IPLR describes the ratio of completely lost packets per total sent packets. The detailed definition of these criteria can be found in [121]. Classes 0-5 specify upper limits for default Internet applications, such as WebTV, and are not suitable for IPTV scenarios. For example, QoS classes 0/1 focus on real-time applications that are sensitive to jitter. They introduce a loss rate of at most 10^{-3} and an upper error rate of 10^{-4} while limiting the maximum delay variation of 50 ms. As QoS class 0 targets highly interactive applications, it narrows the transfer delay to 100 ms, compared to 400 ms in class 1. Classes 2/3 equal the classes 0/1 but skip the requirement for the delay variation. Class 4 equals classes 2/3, but extends the transmission delay to 1 second and thus represents a low loss transmission only. Class 5 focuses on traditional applications and requires best-effort network techniques. Classes 6-7 provide adequate QoS requirements for IPTV applications and constitute the most restrictive classes. They refine QoS classes 0/1 for real-time applications by lowering the loss and error rate by a factor of 100 and introducing a packet reordering ratio (IPRR) of at most 10^{-6}.

In general the underlying network layer is not able to correct errors up to these limits for IPLR and IPER. Application layer forward error correction (AL-FEC) schemes are recommended as presented in [122].

3.7 Exemplary Transmission System: DVB-IPTV

The *DVB Project*[5] provides a set of specifications for *Internet Protocol TV (IPTV)* transmissions as introduced in section 3.1.2. The specification for DVB-IPTV primarily focuses on the distribution of IPTV via a logical interface to the home network. It does not target the specification of a complete end-to-end IPTV architecture. It is possible to integrate the specification into an architecture as specified with ETSI TISPAN [123, 124] and proposed in the DVB BlueBook A 128 [125].

IPTV is generally implemented as a subscription-based service involving multiple different partners and domains. Figure 3.12 depicts a coarse layer model of DVB-IPTV,

[5] http://www.dvb.org (2015/01/29)

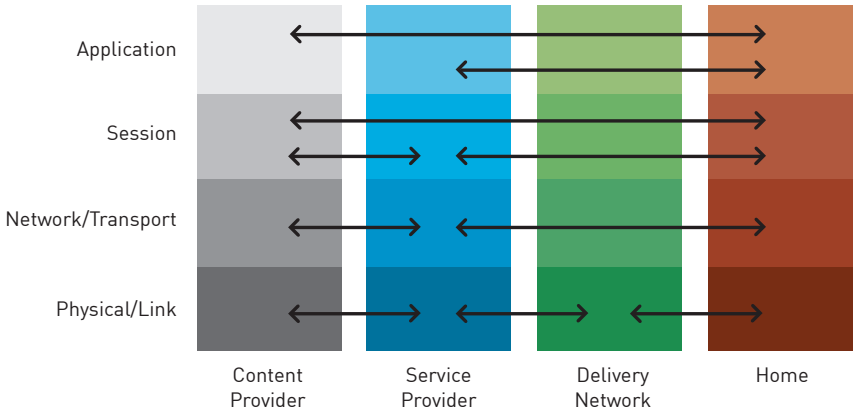

FIGURE 3.12: The DVB-IPTV domains.

including all these domains and arrange them according to the ISO/OSI reference model described in ITU-T recommendation X.200 [38]. The figure also shows the communication paths between the different domains with respect to the corresponding OSI layer.

The illustration identifies four different domains and partners:

- **Content Provider:** The content provider generates the content and provides it to customers along with some metadata. In general the user at home has a contract with this provider and accesses content directly from the content provider.
- **Service Provider:** The service provider may license content from a content provider, packs this content into an IP format and offers it as a service.
- **Delivery Network:** This network connects user and service provider. The delivery network consists of a core and an access network and may use a set of different technologies that are transparent to the IP packets. It is maintained by a network provider and can run different *Quality of Service* mechanisms to ensure a reliable transport.
- **Home:** This represents either a single device or a home network able to receive and display DVB-IPTV content.

DVB-IPTV defines only the *IP Infrastructure (IPI)*-1 interface. This interface logically connects the home devices to the network. The IPI-1 interface also serves as a broker over which all network, transport, session, and application layer protocol interactions occur [122].

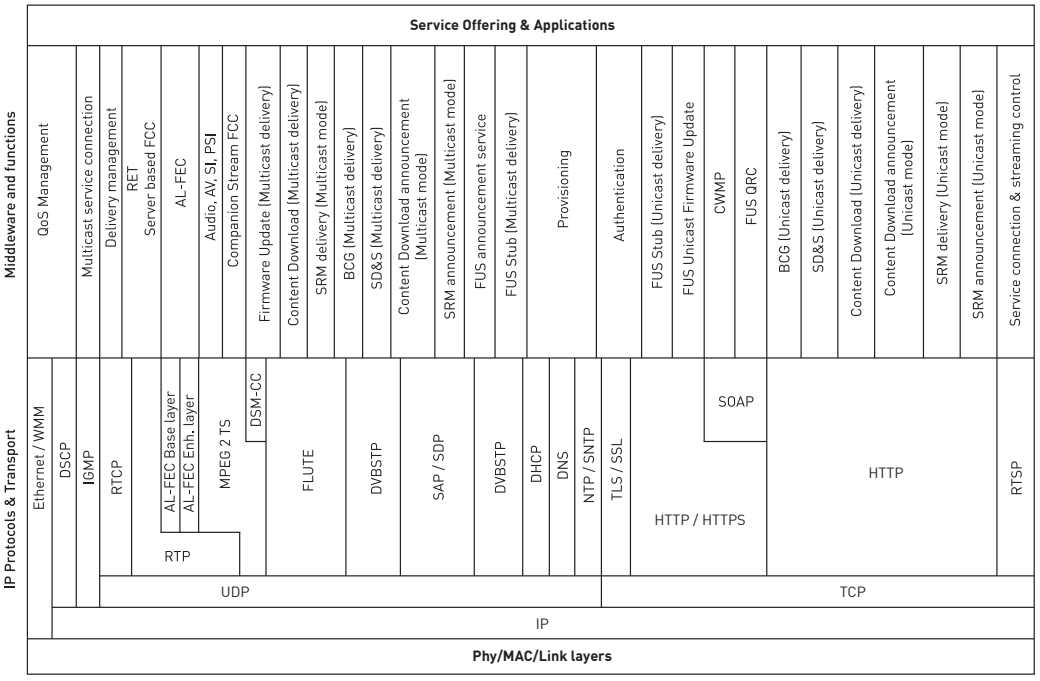

FIGURE 3.13: The DVB-IPTV protocol stack.

Figure 3.13 presents an informative overview of the complex DVB-IPTV protocol stack. The stack structure basically follows the ISO/OSI model convention. It can be grouped into four individual parts:

- **Service Offering:** This is the topmost layer. It includes all application specific aspects to create a DVB-IPTV service application on top of an IP network.
- **Middleware and Functions:** This layer describes the functionality specified in the core DVB-IPTV standard ETSI TS 102 034 V1.5.1 [122]. *Middleware and functions* additionally include other DVB-related specifications.
- **IP Protocols and Transport:** This layer describes the protocols and transport technologies used by the *Middleware and Functions* layer. It defines how data is transmitted between a source and a sink.
- **Physical Layer:** The protocol stack is independent of the present physical layer. A wide variety of different ISO/OSI physical layer representatives are possible.

DVB-IPTV complies with the principles of *Differentiated Services (DiffServ)* [10]. It is assumed that IP packets are individually marked when crossing the IPI-1 interface. Consequently, the data source sets the specific *Differentiated Services Code Point (DSCP)* [126] flag in the packet's header.

There are five categories that classify the content and its handling in the network: *voice*, *real-time voice*, *real-time video*, *voice and video signaling*, and *best effort*.

The individual DSCP values can be found in RFC 2474 [111]. The groups are used for both networking and data-link layer, where either the *type-of-service (TOS)* field in the IP header or tags according to IEEE 802.1Q [127] are used.

Reliability is provided by the application of two error correction codes: a simple *packet-based interleaved parity code* and a *Raptor code*. The former code should follow the standard proposed by the *Society of Motion Picture and Television Engineers (SMPTE)*[6] ST2022-1:2007 [128] and ST2022-2:2007 [129]. The Raptor code [130] is based on an *FEC Streaming Framework* and *FEC Schemes* according to RFC 5052 [131]. Details for the proposed Raptor code can be found in chapter E.7 of [122].

[6] http://www.smpte.org (2015/01/29)

Chapter 4

Transport Optimization

Optimization is a task of adapting a specific situation to its environment. The adaptation must achieve an optimal fitting depending on a set of requirements. Such a task generally has *fixed* and *variable* parameters, that reflect the current status. Optimum results are obtained with *optimization*, where *a set* of values for the variable parameters is efficiently selected. During that process the fixed variables are used as a side information. *Optimum* in this context is a coarse description. It highly depends on the given situation and environment as well as the targeted operation. In the domain of telecommunications, *optimization* can imply the selection of an optimum route in a network, considering resource utilization.

The process of finding an adequate solution is bounded by a set $\mathcal{Z}$ of so called *objective functions* Z_i as shown in equation (4.1).

$$\mathcal{Z} = \{Z_1, Z_2, Z_3, \ldots\} \tag{4.1}$$

The number of given objective functions for an optimization task directly corresponds to the task's complexity. A larger number of optimization goals generally implies the usage of more objective functions. The feasible solution depends on the combination of all objective functions.

An objective function can be defined as in equation (4.2).

$$Z : X \mapsto Y \subseteq \mathbb{R} \tag{4.2}$$

X can represent any kind of element, such as characters, numbers, arrays, matrices, etc., as long as the objective function correctly utilizes the input to generate the output value Y. The precise way of calculating this output value depends on the given environment and cannot be generally defined.

Optimization means to find a set of input values x_i, that generates the optimum output with respect to the given objective functions Z_i. Depending on the optimization task, x_i is of any dimension, e.g. $x_i \in \mathbb{R}^n$ with $n \geq 1$.

The difference between the task of *optimization* and a *trivial solution lookup* is significant. When searching a solution, the knowledge about the solution is already present. This implies that the search process just checks if the set of input $x \in X$ is feasible and represents a solution for the problem. With optimization, there is no knowledge about the solution and the process tries to find a feasible set of inputs $x \in X$. The objective functions Z thereby judges the quality of the x. Consequently, search algorithms are a subset of optimization processes.

The common part of all optimization approaches is to formulate a precise abstraction model of the present problem and its environment. It is required to use an adequate representation that reflects important characteristics, while skipping non-relevant information to decrease the complexity of the model.

Optimization approaches can be categorized regarding the knowledge about the solutions. In case the solution characteristics are perfectly known, it is suitable to differentiate between *feasible* and *non-feasible* solutions. In this case, it is also known how adjacent solution candidates behave with respect to their global utility. The knowledge about the solution space is eventually present and it can be efficiently analyzed. This group of algorithms is called *deterministic algorithms*.

On the other hand, there are *probabilistic algorithms*. They are used in case the quality of a solution candidate is not known or observable. It is possible that adjacent solution candidates significantly differ in their fitness. Probabilistic algorithms are used in case the search space is large and of higher dimension. This approach is often represented by *Monte Carlo algorithms* [132]. These algorithms offer solution correctness by improved run-time. In the majority of scenarios it is suitable to have a *slightly* suboptimal solution after a short period of calculation, instead of having the optimum solution after a long period of calculation. Particularly, real-time applications such as live multimedia streaming benefit from shorter latencies while obtaining solutions close to the optimum.

Optimization of transport structures, such as IP networks, can have multiple different aspects, especially when the transported data has strict requirements.

It is possible to reroute or redirect traffic in the network over segments that are more suitable than others, e.g. multimedia over fast and email over slow links. Another approach is to introduce new metrics that use more than a single information to represent the weight, apply well-known graph theoretic approaches to determine network bottlenecks, or any other traffic engineering mechanism.

New transmission paradigms that *revolutionary* change, or at least extend current approaches, can cause a significant increase in transmission efficiency. This implies that fundamental modifications at network structures can be required. In case traditional structures remain unmodified and new ideas are implemented in present transport network environments, they can be described as *evolutionary*. Both approaches must not be considered individually, but can be perfectly used in combination. The design of an adequate and efficient transport scheme has many facets and depends on the present environmental situation. All approaches focus on the transmission data rate. The number of applications transmitting a large amount of data over the networks is continuously increasing. Underrated spatial characteristics carry an optimization potential for the future media Internet.

This chapter introduces popular optimization schemes, such as *mathematical optimization*, *graph theoretical approaches*, and *metrics*. Each topic is equipped with a representative application example for the network transport domain. An interactive network optimization scheme is presented, that targets direct knowledge exchange between an application and the network infrastructure.

4.1 Optimization Models for Network Environments

The optimization of network infrastructures requires an appropriate model. A network typically consists of *devices* and *links*. A crucial task is to find a mathematical representation of the present topology including all important characteristics. Links typically have *properties* such as latency, maximum data rate, packet loss, or jitter. These parameters are generally reflected by *weights*. Each link in the network model has a corresponding weight. This weight can either consist of a single entry or a set of entries. In the latter case, it is called a multi-dimensional weight. In the domain of network transport, *weights* are often called *metrics*. Nodes are often characterized by their location inside the network. Interesting representative features are the number of connected links or the number of shortest paths traversing a node. In section 6.2.2 these concepts are further presented. Nodes can either reflect a single physical device, or a group of devices including their interconnections. In this case, a node represents a *cloud*. In any case, the granularity of the represented environment depends on the application and the environment itself. Special characteristics as *virtual LANs (VLAN)* [127] or *Link Aggregation Groups (LAG)* [133] must be individually considered and modeled.

4.2 Mathematical Optimization

Mathematical optimization is the process of selecting an element distribution out of a set of potential candidate distributions according to a set of side constraints. *Element distribution* refers to an order of suppliers, an employee's-projects assignment, or a set of links forming a transmission route for data in IP networks. In each case individual element distributions are generally conceivable, but only a single distribution is considered to be *efficient* according to some requirements and weights, represented in the side constraints. Mathematical optimization generally describes and solves a *minimization* or *maximization* problem. The objective function, as introduced in equation (4.2), and the side constraints are formulated as mathematical functions. These functions have limited characteristics according to the specific type of mathematical optimization. A popular subcategory of mathematical optimization is *Integer Programming (IP)*[1] [134].

In this thesis *Linear Programming (LP)* [135] is frequently used to model network topologies and appearing optimization tasks. LP is a special type of *Convex Programming* where all functions are exclusively built by linear combinations of variables. The most prominent algorithm to solve *Linear Programs* is the *Simplex Algorithm* [135].

4.2.1 Linear Programming

Linear Programming describes an optimization approach that consists of three main components. Issues to be optimized are represented by *decision variables* x_i. The values of these variables are bounded to special ranges depending on the problem type. In case the model is *Integer Programming (IP)* it holds $x_i \in \mathbb{N}_0$. Linear Programming uses decision variables $x_i \in \mathbb{R}_0^+$ with a continuous value space. There also exist variants that restrict the values to binary values or a combination of different domains. In this case, the optimization approach is called *Integer Linear Programming (ILP)* [134], *Binary Integer Programming (BIP)*, or *Mixed Integer Linear Programming (MILP)* [135]. As an example, *BIP* fixes the co-domain to $\{0, 1\}$. In case non-continuous values for the decision variables are allowed, the problem is classified as NP-hard. For example, the BIP is listed in *Karp's 21 NP-complete problems* [136].

The value of x_i indicates the utilization of the corresponding issue.

For example, in a BIP the decision variable's value $x_i = 0$ implies that the corresponding object is not considered in the global solution. In case $x_i = 1$, the BIP assumes the linked object as a part of the global optimum solution. Multiple decision variables are represented as a vector $\mathbf{x}$, as shown in equation (4.3).

[1] Do not confuse with the *Internet Protocol (IP)* as defined in RFC 791 [1]

The *objective function* of the LP model, as shown in equation (4.3), is expressed as a linear function $z(\mathbf{x})$ depending on the decision variables and some constants c. It represents the core quantity that is to be optimized.

$$\begin{aligned} z(\mathbf{x}) &= c_1 \cdot x_1 + c_2 \cdot x_2 + \ldots + c_n \cdot x_n \\ &= \sum_{i=1}^{n} c_i \cdot x_i \\ &= \mathbf{c}^T \cdot \mathbf{x} \end{aligned} \tag{4.3}$$

The objective function is generally to be *maximized* or *minimized*.

The optimization *constraints* are a composition of linear equalities or inequalities. The weights a for each object are multiplied with the corresponding indicator variables and finally accumulated. This weighted sum is bounded by a threshold b that constitutes the given constraint. The standard form of a *Linear Program* is shown in equation (4.4).

$$\underbrace{\begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \ldots & a_{mn} \end{pmatrix}}_{=:A} \cdot \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \leq \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix} \tag{4.4}$$

A complete LP in standard form with m constraints and n variables is presented in equation (4.5).

Objective function:

$$\max \quad z(\mathbf{x}) = \sum_{i=1}^{n} c_i \cdot x_i \tag{4.5}$$

subject to:

$$\sum_{i=1}^{n} a_{j,i} \cdot x_i \leq b_j \qquad \forall 1 \leq j \leq m \tag{4.6}$$

The LP representations from equation (4.5) can be shortened to:

Objective function:

$$\max \quad z(\mathbf{x}) = \mathbf{c}^T \cdot \mathbf{x} \tag{4.7}$$

subject to:

$$A \cdot \mathbf{x} \leq \mathbf{b} \tag{4.8}$$

Linear Programs can be solved with the *Simplex Algorithm* [135]. LPs are well suited to model network topologies and to find optimum routes. In the following a set of sample LP formulations is presented that finds efficient unicast, multicast, and broadcast routes in a generic network.

4.2.1.1 Unicast Network Routing Model

The unicast routing problem formulation as an LP model represents the basis for further optimization models derived in this thesis.

The main task is to efficiently route a single stream through a network. For this problem a set V is defined that includes all vertices of the network. A set E is defined that contains the edges (i, j) between any vertices (n_i, n_j). In addition to E, the sets E_i^+ and E_i^- are defined that include all edges terminating and originating at node n_i, respectively.

The decision variables for the LP model are $x_{ij} \in 0, 1$. Each x_{ij} reflects if the corresponding edge (i, j) is active or not. An edge (i, j) has a weight w_{ij} as well as a capacity c_{ij}. The weight reflects the quality of the edge and is used in the optimization process to select the most efficient route, i.e. the one with minimum weight, in the network. The capacity c_{ij} represents the upper data rate limit of an edge (i, j).

It is important for the model to strictly differentiate between *source*, *sink* and *intermediate node*. Equation (4.9) defines the variable b_i. It identifies the type of the corresponding node n_i.

$$b_i = \begin{cases} 1 & \text{, node } n_i \text{ is a source} \\ -1 & \text{, node } n_i \text{ is a sink} \\ 0 & \text{, otherwise} \end{cases} \tag{4.9}$$

The value of b_i reflects the amount of *flow* processed at a node n_i. A source is *generating* flow, a sink is *consuming* flow, and an *intermediate node* is solely forwarding flow.
Each data stream in a network is assumed to have a positive average data rate d. If a

stream data rate d is traversing an edge (i, j), this edge is loaded with d and thus its current effective capacity is reduced by d.

The present unicast flow routing problem is concerned with the selection of an optimum path in the network. Two requirements can be identified. Firstly, the data generated at the source must reliably reach the designated sink. This means that only the source is generating data and the sink is solely consuming data of this flow. The remaining nodes only forward incoming data to another node as already defined in equation (4.9). Consequently, *flow* can not arbitrarily vanish or appear in a network.
This requirement is known as the *flow balance* [25] and represented in the LP model as shown in equation (4.10).

$$\forall i \in V : \sum_{(i,j)\in E_i^-} x_{ij} - \sum_{(i,j)\in E_i^+} x_{ji} = \begin{cases} 1 & \text{, node } n_i \text{ is a source} \\ -1 & \text{, node } n_i \text{ is a sink} \\ 0 & \text{, otherwise} \end{cases} = b_i \tag{4.10}$$

Additionally, the selected path must be able to handle the data generated at the source. The additional data rate d on a link (i, j) must not exceed the link's capacity c_{ij}. Equation (4.11) shows the corresponding LP model formulation.

$$x_{ij} \cdot d - c_{ij} \leq 0 \tag{4.11}$$

The optimum path in the network is achieved by selecting the path with the minimal accumulated weight. Consequently, the objective function z can be defined as presented in equation (4.12).

$$z = \sum_{(i,j)\in E} w_{ij} \cdot x_{ij} \tag{4.12}$$

Obviously, only active edges contribute to the accumulation as $x_{ij} = 0$ for non-active edges.

Equations (4.10), (4.11), and (4.12) formulate an LP model that handles unicast routing in networks. This model can be further generalized, as shown in equations (4.13) - (4.16), in order to find an optimum distribution for k concurrent unicast flows [137].

Objective function:

$$\min \quad \sum_{(i,j)\in E} \sum_{k} w_{ij} \cdot x_{ij}^{k} \tag{4.13}$$

subject to:

$$\sum_{(i,j)\in E_i^-} x_{ij}^k - \sum_{(j,i)\in E_i^+} x_{ji}^k = b_i^k \qquad \forall k,\ \forall i \in V \tag{4.14}$$

$$b_i^k = \begin{cases} 1, & \text{if } i \text{ is the source of flow } k, \\ -1, & \text{if } i \text{ is the sink of flow } k, \\ 0, & \text{otherwise.} \end{cases} \qquad \forall k,\ \forall i \in V \tag{4.15}$$

$$\sum_{k} x_{ij}^k \cdot d^k - c_{ij} \leq 0 \qquad \forall (i,j) \in E \tag{4.16}$$

4.2.1.2 Multicast Network Routing Model

The multicast network routing model has an analogous structure as the unicast model. It comprises a single source, but multiple sinks. These sinks are also called terminal nodes and represented by the set T. x_{ij} represents the indicator variable of an edge (i, j), but is accompanied with a new binary variable $y_{ij} \in \{0, 1\}$. A multicast distribution tree contains a root, i.e. the data source, and multiple leafs, i.e. the $|T|$ sinks. The unicast scenario assumes that in case an edge (i, j) is used, it generally transports the amount of flow according to the number of sinks reachable via this edge. For multicast considerations it is important to normalize this value to one flow, as the source sends a single data stream only, independent of the number of listening sinks. Hence, y_{ij} represents an edge (i, j) that is used in the multicast distribution tree. Consequently, if and only if $y_{ij} = 1$, the edge (i, j) is used in the multicast dissemination tree to reach at least one data sink.

This can be modeled in an LP by the help of two new constraints represented in equations (4.20) and (4.21). Equation (4.20) cares for the case that $x_{ij} = 0$ and equation (4.21) handles the upper limit for y_{ij}. This new indicator variable y_{ij} is used in the *objective function* in equation (4.17) and in the capacity constraint in equation (4.22). The flow balance constraint in equation (4.18) is not affected by this modification as it requires the full amount of flow on the links. Otherwise the model is not solvable. In case k concurrent multicast flows must be distributed in a network, the corresponding LP model can be defined as shown in equation (4.17) - (4.22) [137].

Objective function:

$$\min \quad \sum_{(i,j)\in E} \sum_{k} w_{ij} \cdot y_{ij}^{k} \tag{4.17}$$

subject to:

$$\sum_{(i,j)\in E_i^-} x_{ij}^k - \sum_{(j,i)\in E_i^+} x_{ji}^k = b_i^k \qquad \forall k, \forall i \in V \tag{4.18}$$

$$b_i^k = \begin{cases} |T^k|, & \text{if } i \text{ is the source of flow } k, \\ -1, & \text{if } i \text{ is a terminal node of flow } k, \\ 0, & \text{otherwise.} \end{cases} \qquad \forall i \in V \tag{4.19}$$

$$x_{ij}^k \geq y_{ij}^k \qquad \forall k,\ \forall (i,j) \in E \tag{4.20}$$

$$\frac{x_{ij}^k}{|T^k|} \leq y_{ij}^k \qquad \forall k,\ \forall (i,j) \in E \tag{4.21}$$

$$\sum_{k} y_{ij}^k \cdot d^k - c_{ij} \leq 0 \qquad \forall (i,j) \in E \tag{4.22}$$

4.2.1.3 Broadcast Network Routing Model

The broadcast network routing model represents a different type of flow distribution model. It focuses on the calculation of a single minimum spanning tree inside the network that connects all participating nodes. Broadcasting generally does not target strict *Quality of Service* requirements as it is used with basic network service applications, e.g. *Address Resolution Protocol (ARP)* [138] or *Dynamic Host Configuration Protocol (DHCP)* [74]. The broadcast network routing model can be defined as shown in equation (4.23) - (4.26) [137].

Objective function:

$$\min \quad \sum_{(i,j)\in E} w_{ij} \cdot x_{ij} \tag{4.23}$$

subject to:

$$\sum_{(i,j)\in E} x_{ij} = |V| - 1 \tag{4.24}$$

$$\sum_{(i,j)\in S^2} x_{ij} \leq |S| - 1 \qquad \forall\, S \subset V \wedge S \neq \emptyset \tag{4.25}$$

$$x_{ij} \in \{0, 1\} \qquad \forall (i,j) \in E \tag{4.26}$$

The objective function in equation (4.23) represents the active edges and their corresponding weights. It implies a minimization of the edge weights in the final edge set used for broadcast connectivity. As broadcasting represents a tree distribution structure, equation (4.24) introduces this requirement by the fact that the number of edges equals exactly the number of nodes $|V|$ minus 1. Equation (4.25) extends equation (4.24) and guarantees a cycle-free tree. It uses all possible sets $S \subset V$ where $S \neq \emptyset$ and ensures that the sum of selected edges in each set S is smaller than $|S|$ to consequently avoid cycles. Equation (4.26) defines the indicator variables to be binary.

4.2.1.4 Joint Unicast and Multicast Routing Model

The joint unicast and multicast routing model deals with the efficient distribution of a composition containing k *unicast flows* and l *multicast flows*, while respecting the individual link capacities. It eventually ends up in the implementation of an optimized stream transmission scheme in the network environment. The $k + l$ flows are assumed to concurrently traverse the network. A unified LP model representing this environment [137] is derived from the basic unicast and multicast routing model. It is presented in equation (4.27) - (4.34).

Objective function:

$$\min \quad \sum_{(i,j)\in E} w_{ij} \cdot \left(\sum_k x_{ij}^k + \sum_l y_{ij}^l \right) \tag{4.27}$$

subject to:

$$\sum_{(i,j)\in E_i^-} x_{ij}^k - \sum_{(j,i)\in E_i^+} x_{ji}^k = b_i^k \qquad \forall k,\ \forall i \in V \tag{4.28}$$

$$\sum_{(i,j)\in E_i^-} x_{ij}^l - \sum_{(j,i)\in E_i^+} x_{ji}^l = b_i^l \qquad \forall l, \forall i \in V \tag{4.29}$$

$$b_i^k = \begin{cases} 1, & \text{if } i \text{ is the source of flow } k, \\ -1, & \text{if } i \text{ is the sink of flow } k, \\ 0, & \text{otherwise.} \end{cases} \qquad \forall k,\ \forall i \in V \tag{4.30}$$

$$b_i^l = \begin{cases} |T^l|, & \text{if } i \text{ is the source of flow } l, \\ -1, & \text{if } i \text{ is a terminal node of flow } l\ , \\ 0, & \text{otherwise.} \end{cases} \qquad \forall l,\ \forall i \in V \tag{4.31}$$

$$x_{ij}^l \geq y_{ij}^l \qquad \forall l,\ \forall (i,j) \in E \tag{4.32}$$

$$\frac{x_{ij}^l}{|T^l|} \leq y_{ij}^l \qquad \forall l,\ \forall (i,j) \in E \tag{4.33}$$

$$\sum_k x_{ij}^k \cdot d^k + \sum_l y_{ij}^l \cdot d^l - c_{ij} \leq 0 \qquad \forall (i,j) \in E \tag{4.34}$$

The objective function (4.27) and the capacity constraint (4.34) are extended by two terms x_{ij}^k and y_{ij}^l representing the k-th unicast and the l-th multicast flow, respectively. Both flow types require a new set of slightly different balance equations (4.28) and (4.29), with different flow values as found in (4.30) and (4.31). The constraints for the indicator variables in equations (4.32) and (4.33) remain unmodified.

As shown in this section, *Linear Programming* serves as a suitable basis for network modeling and optimization.

4.3 Graph Theory

Graph Theory emphasizes the topology information of a given problem. It is mainly used when detailed information is not relevant, but the main structure is considered to be significant. Graph theory generally abstracts problems into a structure including *nodes* and *edges*. The combination of both is called a *graph*. Since its first appearance in 1736, it evolved to a prominent mathematical approach with numerous algorithms and applications [14].

A *graph* can be formally defined as follows [16]:

> A *graph* $G = (V, E)$ consists of a set V containing the vertices or nodes and a set E containing the edges. Each edge $e \in E$ is a tuple of two vertices $e = (u, v)$ with $u, v \in V$. $w(u, v)$ are the weights assigned to each edge of the graph. The *order* of the graph is equal to the cardinality of the set of vertices $|V|$. The *size* of the graph is equal to the cardinality of the set of edges $|E|$. A graph is called *directed* if its edges have a defined orientation, otherwise *undirected*. In case of the directed graphs this means $(u, v) \neq (v, u)$, whereas in the undirected case $(u, v) = (v, u)$ holds.

A basic concept in graph theory is the *weight* of an edge. The weight reflects a comparable quantity that is used by algorithms to decide if an edge is used or not. Weights are normally assigned by metrics that reflect a special property. The most prominent metric is the *hop count*, that represents the number of visited nodes.

The relation between *Graph Theory* and network transport optimization is manifold. In graph theory *paths* [139] and *trees* [139] can be flexibly used to model transmission scenarios in IP networks as they intuitively reflect one-to-one or one-to-many communication scenarios. In those cases the weight of an edge directly corresponds to a quality measure of the connection between two nodes. For instance, this may reflect latency, jitter, or loss probability.

With graph theory, it is also possible to *analyze* networks and discover weaknesses in their topology [139]. Prominent representatives are the *k-vertex connected graph*, *k-edge connected graph*, or *graph diameter* properties.

The mapping from network infrastructures to *graphs* is straightforward. Links correspond to edges and devices correspond to nodes. *Graph Theory* represents a well-known and adequate representation for network optimization.

Compared to *Linear Programming*, it lacks the flexible adjustment for side constraints or objective functions, as in either case the algorithms have to be modified. In case

Algorithm 1 Dijkstra's Algorithm

```
1: procedure DIJKSTRA(G = (V, E),s,w : E → ℝ)
   Input: directed graph G, start vertex s, weight-function w

2:     for each v ∈ V do                                   ▷ initialization
3:         d(v) ← ∞
4:         π(v) ← nil
5:     end for
6:     d(s) ← 0

7:     S ← ∅                                   ▷ set of finished vertices
8:     Q ← V                                   ▷ set of vertices not finished so far
9:     while Q ≠ ∅ do
10:        u ← Extract-Minimum(Q)
11:        S ← S ∪ {u}
12:        for each v ∈ Adjacent(u) do
13:            if d(v) > d(u) + w(u, v) then               ▷ relaxation
14:                d(v) ← d(u) + w(u, v)
15:                π(v) ← u
16:            end if
17:        end for
18:    end while
19: end procedure
```

of an LP model, it is intuitive to use an additional constraint or other elements in the objective function.
In the following, a basic approach for unicast routing in networks is presented.

Single-Source Shortest Path Problem

Assume a graph $G = (V, E)$ and a weight function $w : E \mapsto \mathbb{R}$. The *Single-Source Shortest Path (SSSP)* problem describes the task of finding a shortest path P that connects start node $v_s \in V$ and end node $v_e \in V$ [16].

A path is an ordered set of edges $e_{ij} = (v_i, v_j)$. In case of non-negative and additive edge weights, the weight of P is the sum of weights of participating edges as shown in equation (4.35).

$$w(P) = \sum_{e \in P} w(e) \tag{4.35}$$

P is the shortest path in G from v_s to v_e in case $w(P) < w(P'),\ \forall P' \in G$. *Shortest* in this context refers to the minimum weight sum, that has been accumulated over the selected path. Weights can be either one-dimensional or multi-dimensional, depending on the number of characteristics they represent.

Edsger W. Dijkstra invented an algorithm to solve the SSSP problem [16]. The algorithm assumes a weighted and directed graph $G = (V, E)$ with non-negative, one-dimensional weights. It uses a set of vertices S that contains all already accumulated shortest path weights. In each iteration it selects a vertex $u \in V \backslash \{s\}$ with the smallest shortest path value estimate so far and adds u to S. All egress edges from u are eventually relaxed.

The corresponding pseudo-code of *Dijkstra's Algorithm* is presented in Algorithm 1. Lines 2-8 initialize the algorithm and set up all required variables. The main algorithm starts with the loop in line 9. A vertex u is extracted from Q and inserted into S. Lines 12-16 relaxes all adjacent nodes from u. The algorithm terminates in case Q is empty, that is after exactly $|V|$ iterations.

Dijkstra's Algorithm is used in practical network routing scenarios. It constitutes the calculation framework for the *Open Shortest Path First (OSPF)* [15] routing protocol.

In case multi-dimensional weight vectors are used with legacy graph algorithm, an appropriate mapping function must be defined in order transfer the weight result to $\mathbb{R}$. In the next section metrics and norms are discussed, that can be used for multi-dimensional weighting.

4.4 Metrics and Norms

A metric is a function that calculates the difference between two mathematical objects, such as a real or complex number, a vector, or a matrix. In case one of the two considered objects equals the coordination system origin, the metric yields a value that can be used to rate different inputs. A *metric* cannot be negative. It can be mathematically abstract defined as shown in equation (4.36), where X denotes an arbitrary mathematical object.

$$M : \ X \times X \mapsto \mathbb{R}_0^+ \tag{4.36}$$

A *norm* is mathematically a function that calculates a specific value that represents a single mathematical object. It can be defined to output non-negative values as shown in equation (4.37).

$$\| \cdot \| : \ X \mapsto \mathbb{R}_0^+ \tag{4.37}$$

In the domain of network transmission routing, a metric represents a measurement for quality. In this context, *metric* describes the characteristic of a network ingredient and not the mathematical term. The quality either refers to a complete network, a path, or a single link. In this domain multiple different characteristics are significant, such as loss probability, latency, jitter, or available data rate. A mathematical object can

be defined as an n-dimensional characteristics vector $\vec{c}$, that contains multiple different network characteristics c_i:

$$\vec{c} = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix} \in \mathbb{R}^n \tag{4.38}$$

The application of the *norm* function $\|c\|$ provides a single real value that can be used to compare different vectors $\vec{c}$.

It is important to define a global ordering along the $\mathbb{R}$ domain, such as *greater* or *smaller*. Subsequently, it becomes possible to rate different input vectors according to their metric value. In the multidimensional case, no general means of global ordering exists. Assume two 2-dimensional vectors as shown in equation (4.39) with values $c_{1,1} > c_{2,1}$ and $c_{2,2} > c_{2,1}$.

$$\vec{c}_1 = \begin{pmatrix} c_{1,1} \\ c_{1,2} \end{pmatrix} \text{ and } \vec{c}_2 = \begin{pmatrix} c_{2,1} \\ c_{2,2} \end{pmatrix} \tag{4.39}$$

With this combination it is not possible to order both vectors, as one element of each vector dominates the corresponding one in the other vector. The comparison of multi-dimensional vectors requires a smart mapping to a domain that supports a global ordering. The application of a norm transforms the n-dimensional vector to a single non-negative real value.

Traditional and legacy network applications typically operate with 1-dimensional values, and thus require the application of a norm function. In principal, a lower metric value reflects better characteristics and is preferred over a higher metric value. The concrete relation between two metric values depends on the application. A metric value used for routing in networks must not be negative. In case there are negative values reflecting link metrics, this leads to invalid results as negative values distort the accumulation of the path weight. Special situations emerge with *negative cycles* in networks. Routing algorithms can use these cycles to frequently reduce the overall path metric. The path weight is small whereas the real path length has significantly grown. An exemplary check for negative cycles is shown in algorithm 2. A detailed discussion about negative-cycle check algorithms is given in [140].

Assume the transport of time critical multimedia traffic over an IP network. In order to construct an efficient dissemination structure, an appropriate metric, that includes multiple characteristics of the network segments, is required. Typically distribution mechanisms rely on *Single Source Shortest Path (SSSP)* [16] or *Spanning Tree* algorithms [16] that incorporate a single value, e.g. a minimum hop count. For instance, *Dijkstra's Algorithm* [16] exploits the number of hops to find a feasible path through a

Algorithm 2 Check for negative cycles in networks.

```
1: procedure CHECK-NEGATIVE-CYCLES
2:     for each edge e_ij with weight w(i, j) do
3:         if d(u) + w(u, v) < d(v) then
4:             ERROR: negative cycles found
5:         end if
6:     end for
7: end procedure
```

network. The *Spanning Tree* algorithms of *Kruskal* [16] and *Prim* [16] also employ the number of hops between the sender and receiver.

Real-time multimedia traffic implies an intrinsic time characteristic and the length of a transmission path loses its primary significance. Delivery time and the residual loss probability become more important and finding an optimal network path relies on a multi-dimensional characteristics vector. If two individual characteristics are used, there is no global and unique ordering. A metric must be used to transform the higher-dimensional space into a 1-dimensional space to obtain a higher transmission efficiency.

In the multimedia transmission domain the main task is to *efficiently* transmit audio-visual data from the source to the sink. The network infrastructure provides a potentially large set of nodes interconnected by numerous links. Consequently, there are multiple different possible paths P. A path represents an individually ordered sets of links that connect two end nodes.

The focus is on the selection of an efficient path, according to the network characteristics and the multimedia application demands. The term *efficient* must be defined in this context. Best-effort transmission defines *efficient path* as *shortest path* and uses a 1-dimensional characteristic represented by the number of hops. Hence, the most efficient route equals the shortest route. This is the path with the smallest number of links. As a best-effort service is predominantly available in wide area networks, the term *length* is generally used as a substitute for *efficiency*, even if it is used with other metrics than number of links.

As a best-effort approach is not adequate to multimedia transmissions, more network details must be considered to efficiently transmit this kind of data. Consequently, a higher-dimensional characteristic space must be exploited where each link is represented by a set of characteristics as shown with the vector $\vec{c}$ introduced in equation (4.38). Following the general nomenclature, the efficiency or length L_0 of a path P that contains

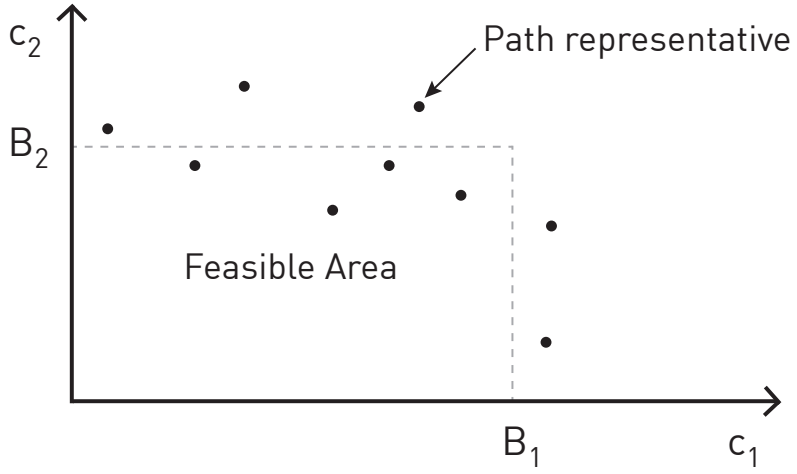

FIGURE 4.1: Feasible area for 2-dimensional optimization.

m links can be generally defined as the sum over all link's characteristics l_i as shown in equation (4.40).

$$\begin{aligned} L(P) &= \sum_{\text{links } l_i} \vec{c_i} = \sum_{\text{links } l_i} \begin{pmatrix} c_{i,1} \\ c_{i,2} \\ \vdots \\ c_{i,n} \end{pmatrix} \\ &= \begin{pmatrix} c_{1,1} + c_{2,1} + \ldots + c_{m,1} \\ c_{1,2} + c_{2,2} + \ldots + c_{m,2} \\ \vdots \\ c_{1,n} + c_{2,n} + \ldots + c_{m,n} \end{pmatrix} \\ &= \begin{pmatrix} c_{P,1} \\ c_{P,2} \\ \vdots \\ c_{P,n} \end{pmatrix} \end{aligned} \tag{4.40}$$

Each path P is defined by its length $L(P)$ and can be represented in an n-dimensional search space. Figure 4.1 illustrates such a search space for $n = 2$ where each point corresponds to an individual path. As multimedia transmissions introduce constraints for each characteristic, B_1 and B_2 define the borders of a rectangular *feasible area* within the global space, built by element domain c_1 and c_2. In case a point lies within the feasible area, all elements satisfy the constraint requirement. Otherwise, at least one element exceeds a constraint. Note that with more dimensions the feasible area becomes a feasible space.

Name	Definition
Manhattan Norm	$\|\vec{x}\|_1 = (\sum_{i=1}^{n} \|x_i\|)$
Euclidean Norm	$\|\vec{x}\|_2 = \sqrt{(\sum_{i=1}^{n} \|x_i\|^2)}$
Maximum Norm	$\|\vec{x}\|_\infty = \max_{i=1..n} (\|x_i\|)$

TABLE 4.1: List of the most popular p-norms: Manhattan, Euclidean, and Maximum norm.

New routing algorithms should potentially be able to find routes based on multiple characteristics, e.g. by element-wise comparison with prioritization. This leads to revolutionary routing approaches. Routing with multiple additive constraints is known to be NP-complete [141] and thus not practical. As evolutionary approaches focus on the seamless integration into currently used technology, it is required to find a mapping from n-dimensions to exactly *one*-dimension. Traditionally used routing algorithms, such as Dijkstra's Algorithm [16], or the spanning tree algorithms of Kruskal and Prim [16], use 1-dimensional search spaces.

In [142] a set of routing metrics and constraints for *Routing Protocols in Low-Power and Lossy Networks (RPL)* [143] are specified.

Despite the existence of multiple available norms, it is crucial to be aware of the implications for routing algorithms. As the characteristics are contained in a vector, the focus is on general vector norms. A vector norm must fulfill the following properties [144]:

$$\begin{aligned} \|\vec{x}\| &> 0 && ,\forall \vec{x} \neq \vec{0} \\ \|\alpha \cdot \vec{x}\| &= |\alpha| \cdot \|\vec{x}\| && ,\alpha \in \mathbb{R} \\ \|\vec{x} + \vec{y}\| &\leq \|\vec{x}\| + \|\vec{y}\| \end{aligned} \tag{4.41}$$

The most commonly used norms are p-norms [145]. The general p-norm is defined as:

$$\|\vec{x}\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}} \tag{4.42}$$

In practice, norms with $p = 1$ (*Manhattan Norm*), $p = 2$ (*Euclidean Norm*), and $p = \infty$ (*Maximum Norm*) are frequently used. Table 4.1 presents these norms in more detail.

Combinations of the presented norms are also valid. As an example, a combined five-dimensional metric could be defined as shown in equation (4.43).

$$\|\vec{x}\| = \pi \cdot |x_5| + \sqrt[3]{|x_1|^3 + |x_3|^3 + |x_4|^3} + \max(|x_1|, |x_2|) \tag{4.43}$$

The eventually used norm depends on the application scenario and the given characteristics. It must be individually defined for each case.

The elements of the achieved path efficiency $L(P)$ can also be adjusted by multiplication with a weighting vector $\vec{w}$:

$$L'(P) = \vec{w} \cdot L(P) = \begin{pmatrix} w_1 & w_2 & \dots & w_n \end{pmatrix} \cdot L(P) \tag{4.44}$$

This may be required in order to scale the vector entries according to given requirements. For the following an already weighted vector $L(P)$ is assumed and does not contradict the assumptions given in equation (4.41).

The adequate selection of a suitable norm for constraint search space is highly important.

In figure 4.1 each point represents an individual path with element-wise accumulated characteristics. The application of a norm transforms these characteristics to a single non-negative real value. An algorithm searches the most efficient path, and thus has to scan the search space. Consider the case where *efficient* is linked with a small norm value. Abstractly, the algorithm starts with a reference value equal to zero and gradually increases this reference value until it hits the first point. This point represents the path with the smallest length, and thus the highest efficiency. In a constraint multi-dimensional search space, this process requires sophisticated norms. In [141] DeNeve proposed two approaches along with a *Tunable Accuracy Multiple Constraints Routing Algorithm (TAMCRA)* to optimize the multi-constraint routing when using Dijkstra's Algorithm. The focus is on the reduction of complexity, when searching a path according to n constraints. The application of a *norm* reduces the problem to a 1-dimensional space. These approaches are presented and discussed next.

4.4.1 Manhattan Norm

The *Manhattan norm* can be applied in case all n available vector elements have equal linear bases. That means all characteristic elements are in the same order of magnitude. The basic method is to create a linear combination:

$$z = \|\vec{c}\|_1 = \sum_{i=1}^{n} |w_i| \cdot |c_i|. \tag{4.45}$$

w_i represents the weight for the characteristic element c_i as described in equation (4.44). It can be seen, that multiple inputs of c_i may result in the same metric value z.

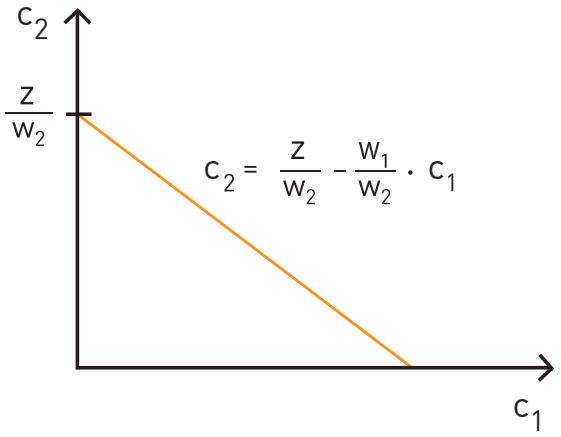

FIGURE 4.2: Norm's slope behavior for two characterization elements.

Assume the values of z and c_j to be fixed. Then, equation (4.45) can be transformed to the form

$$c_i = \frac{z}{w_i} - \sum_{\substack{j=1 \\ j \neq i}}^{n} \frac{w_j}{w_i} \cdot c_j. \tag{4.46}$$

In order to determine the required values of c_i that lead to the value of z, all other elements c_j must be fixed.

For the case $n = 2$, this mapping represents a strictly monotonically decreasing line as shown in equation (4.47).

$$c_2 = \frac{z}{w_2} - \frac{w_1}{w_2} \cdot c_1 \tag{4.47}$$

It can be intuitively drawn as shown in figure 4.2. The corresponding slope depends on the selected value of z, as well on the weighting factors w_1 and w_2. The construction of the function $c_2 : c_1 \mapsto \mathbb{R}$ also directly influences the scanning process of the feasible area, as described in the following.

For example, consider Dijkstra's Algorithm [16], that looks for a minimum path length z and uses the number of segments to represent this length according to equation (4.45). It starts with $z = 0$ as this represents the lowest valid allowed path length, and as only non-negative lengths are assumed. The algorithm implicitly increases the value of z gradually, until a path representative, i.e. a point in the search space, is hit. This algorithm indeed does not consider feasible areas in the search space and eventually assumes all found results to be valid. In the 2-dimensional case, an increasing z indicates a parallel shift of the function line defined in equation (4.47) along the positive c_1 axis. When it hits a point in the search space, this point represents a path with a specific length z. Figure 4.3 illustrates this process and indicates three lines for different values of z.

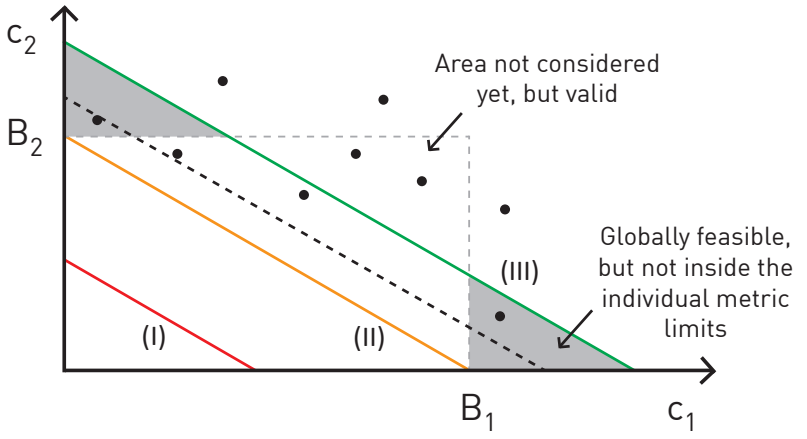

FIGURE 4.3: Manhattan norm approach.

Line `(I)` lies inside the feasible area and has a small value of $z/w_2 < B_2$. For line `(II)`, it holds $z/w_2 = B_2$. It can be seen, that the value of z has been increased and consequently, the functions line has departed from the origin. Line `(III)` represents $z/w_2 > B_2$. In case a point is hit, its value may be minimal, but as the corresponding functional line scans a significant part outside the feasible area, a secondary check is required to avoid non-feasible path representatives.

The first hit, as indicated by the dashed line, is a point that has the smallest value found so far. But in this scenario, it lies outside of the feasible area. The search algorithm is unaware of the fact that the found entry is invalid as $c_2 > B_2$, although $c_1 < B_1$.

This example shows that the slope, defined by the weighting, is a critical part of the norm. It adjusts and influences the search behavior significantly. A smaller slope $-\frac{w_1}{w_2}$ corresponds to either increasing w_1 or decreasing w_2 while fixing the other weight. In this case, the parallel shift earlier hits paths with dominating c_1 elements. Analogously, a larger slope leads to a higher hit significance for elements c_2.

Let B_1 and B_2 be the constraints that set up the feasible area. In order to avoid the generation of invalid hits, the weights must fulfill the condition shown in equation (4.48). This weight ratio directly corresponds to the slope of the lines in figure 4.3.

$$\frac{w_2}{w_1} = \frac{B_1}{B_2} \tag{4.48}$$

It holds $w_1 = 1/B_1$ and $w_2 = 1/B_2$.
For each hit with $z \leq B_2$ it is guaranteed to have found a point inside the feasible area. In case $z > B_2$, an additional check is required to validate the found point in the search

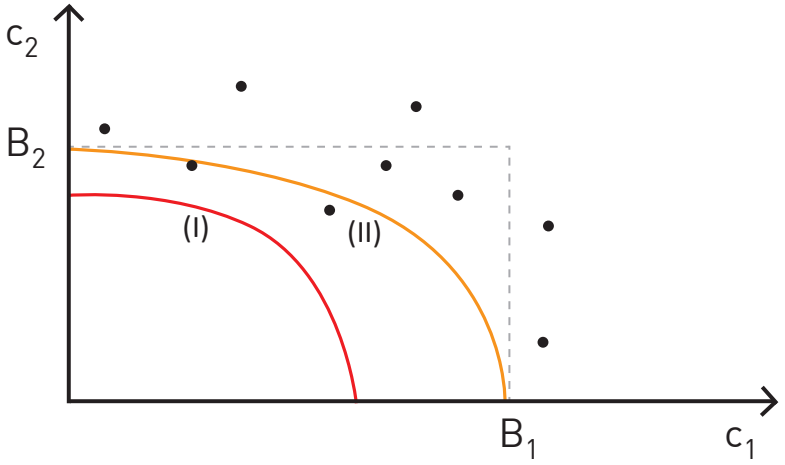

FIGURE 4.4: Euclidean norm approach.

space. Figure 4.3 shows lines with exactly this slope. This process can also be extended to high dimensions.

The composite norm should output the values that are within the feasible area. With the *Manhattan* norm, the algorithm must stop shifting the line as soon as the line crosses the diagonal of the feasible area. Everything beyond this point can produce wrong decisions. In case the applied algorithm cannot be adapted to check the feasibility of the metric outcome, e.g. since the algorithm source is not available, half of the feasible region could be left unscanned.

4.4.2 Euclidean Norm

The *Euclidean norm* reduces the size of the unscanned feasible area, as it uses curved lines instead of straight lines for the reference value z. Consequently, it better approaches the feasible area boundaries introduced by the constraints B_i.

The *Euclidean norm* can be defined as [141]:

$$\begin{aligned} z = \|\vec{c}\|_2 &= \sqrt{\sum_{i=1}^{n} \left(|w_i| \cdot |c_i|\right)^2} \\ &= \sqrt{\sum_{i=1}^{n} \left(\frac{|c_i|}{|B_i|}\right)^2} \end{aligned} \tag{4.49}$$

Equation (4.49) represents the *Holder's q-Norm* [145] and is of significant importance in the theory of *Banach spaces* [144].

Figure 4.4 presents the application of equation (4.49) for $n = 2$. It becomes clear that the feasible area is approached more efficient as with the *Manhattan norm*. Optimally, the lines with the reference value z should perfectly match the feasible area edges. This implicates that the application would result only valid hits.

Considering equation (4.42) and increasing $p \to \infty$ yields this behavior. The resulting norm is known as the *Maximum norm* [141]. This norm is of minor importance for practical multimedia routing applications, as it results the most significant element of the characteristics vector, whereas the remaining elements stay unexploited.

4.5 Network Supported Congestion Avoidance

The *Network Supported Congestion Avoidance (NSCA)* mechanism refers to an *active* and *revolutionary network collaboration* approach [R3]. Optimizing network traffic optimally includes an information exchange between the data source and the transport infrastructure. The granularity of the information directly corresponds to the resulting transmission efficiency, since a detailed set of requirements, as well as demands, eases the optimization process and enables a precise tuning of the infrastructure parameter. An efficient transport consequently includes a knowledge exchange between both entities.

In case current networks try to optimize the transport of data packets, they predominantly *guess* at best the characteristics and requirements of the corresponding data streams. These guesses are based on objective transmission parameters, such as IP addresses or transport ports. Based on this assumption, a potentially suitable path is selected and the transmission initiated. The data applications have no knowledge about the network. They reactively adapt their transmission behavior to the observed environment, e.g. by congestion avoidance mechanisms as done with TCP [3]. The transportation is based on *implicit* knowledge.

Transmitting multimedia data is a challenging task since both, application and network, have to be linked somehow in order to obtain a satisfying *Quality of Experience (QoE)* [9] for the end user. In this section, an interactive approach for exchanging information is proposed, that enables an efficient multimedia transmission. The approach can be applied to any size of transport networks, and can also be interpreted as a basic caller admission scheme [146].

Therefore, a UDP [2] communication protocol is proposed that is used to exchange information about *requested*, *available* and *already occupied* data rates between a data source and the transport infrastructure.

The three main consequences are:

- The *network* is able to perform an improved forwarding strategy based on the requested and occupied data rates.
- The *receiving device* is not obliged to care for congestion controlling, as it is completely shifted to the responsibility of the network.
- The *sending device* must be able to forward information to a controlling entity, that represents the network infrastructure and has administrative privileges.

The individual message exchange is classified as an *active* network collaboration approach and the subsequently initiated transmission is therefore based on *explicit* knowledge.

Traditional networks do not inherently provide a generic way to communicate with the network itself. It is always required to contact a special service listening on an individual address that has administrative maintenance privileges on a set of devices in the network. This approach requires both, *active and passive* functionality, such as setup forwarding rules as well as gathering statistics.

The presented *Network Supported Congestion Avoidance (NSCA)* approach includes three parts:

- *Signaling* of information between device and network.
- *Workflow* descriptions for the device and the network.
- A *Network Rate Control Service* as part of the optimization process.

NSCA is a revolutionary approach as it requires modifications at both network and end-device applications to exchange explicit knowledge. In this context, *Software Defined Networking (SDN)* [21] environments are considered to be flexible to build the network infrastructure for NSCA.

4.5.1 Protocol Specification

The signaling procedure is based on UDP packets. The specified protocol is used to exchange information about *requested*, *available*, and already *occupied* data rates for both, the application data source and network infrastructure. Figure 4.5 shows the signaling procedure between a data source and the network. The network evaluates the

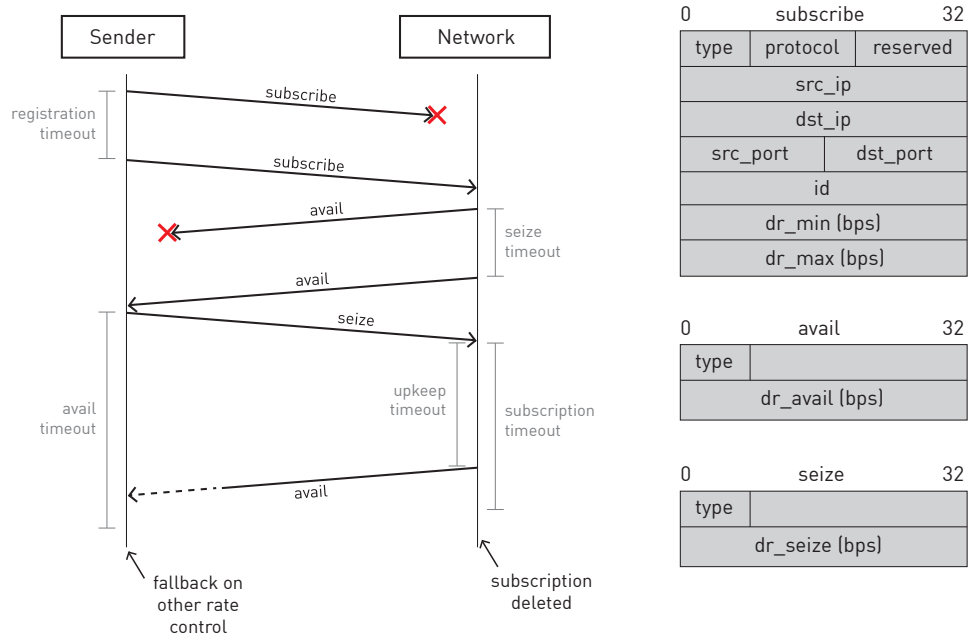

FIGURE 4.5: NSCA signaling process and message prototypes.

received NSCA protocol packets and performs actions as described in section 4.5.2. As a practical example, the responsible network entity is represented by an SDN controller. Then, the corresponding SDN nodes are configured to directly forward packets addressed to a special service address. Generally, it is important to reserve a special address, e.g. $\text{IP}_{NSCA} : \text{Port}_{NSCA}$, to avoid confusions with real existing devices connected to that network.

The data source starts the communication with a `subscribe` message to the network specific service address $\text{IP}_{NSCA} : \text{Port}_{NSCA}$. This message includes information about the *minimum and maximum data rates* that the source can offer, as well as a source individual unique *id*. It additionally includes an OSI layer 3 and 4 flow description of the stream that is to be sent. The source implicitly requests regular notifications about the available data rate in the network by sending this `subscribe` message.

Subsequent to the reception of the `subscribe` message, the network responds with an `avail` message. The `avail` message confirms the currently available data rate in the network to the requesting data source.

After the reception of an `avail` message at the source, it checks the proposed data rate. In case the source agrees with the proposal, it finally sends a `seize` message to the network. This message includes the data rate that the source is going to use. The `seize` message represents the official confirmation to the network.

A reliable and up-to-date message handling also requires multiple timeouts on both communication sides as illustrated in figure 4.5. The data source maintains timeouts for two events, i.e. the *registration* and the *reception* of `avail` messages. The *registration* timeout to_{reg} is started after a `subscribe` message has been sent to the network. to_{reg} expires in case no responding `avail` message is received. With the reception of an `avail` message the corresponding avail timeout to_{avail} is started. This timeout is used to ensure that the information of the available data rate in the network is recent. Since `avail` messages are frequently sent by the network, each received `avail` message restarts to_{avail} at the data source. In case no `avail` message is received, the timeout expires and the source assumes the NSCA service to be unresponsive. In this case, the data source must fall back to a default congestion control mechanism such as *TCP Friendly Rate Control (TFRC)* [147]. Even if NSCA is not responsive, the sender should continuously attempt to contact the service again by repeatedly sending `subscribe` messages to $\text{IP}_{NSCA} : \text{Port}_{NSCA}$. This workflow ensures an early and reliable notification in case the NSCA service restarts.

The network site also maintains multiple timeouts. The *subscription* timeout to_{sub} is managed by the network for each *active* data sink identified by its *sender id*. Every `subscribe` and `seize` message refreshes the corresponding to_{sub} timeout. In case to_{sub} expires, the associated subscription is regarded as inactive and the corresponding data source removed from internal considerations. The *seize* timeout to_{seize} manages a reliable exchange of available data rates. It is started every time an `avail` message is sent and in case no `seize` message has arrived before expiration, the `avail` message is sent again. With the reception of a `seize` message the network starts an *upkeep* timeout to_{upkeep}, that initiates a new *avail/seize* cycle in case the corresponding timeout to_{upkeep} has been expired.

4.5.2 Network and Client Workflow

Besides the basic protocol description an associate workflow specification is required for efficient results. Figure 4.6 presents the NSCA workflow of the *network* as a compact chart. Note, that *network* generally describes an instance, that handles all events and operations on behalf of the network. In case of Software Defined Networking environments, as assumed here, this part is performed by the SDN controller.

The network controller maintains a cache for currently active *subscriptions* and corresponding flows in the network. This cache can be used to calculate current data rates on a per-link basis. The *setup* process for optimization is either triggered by an event or on a regular basis. After initialization the network control flow directly steps into *schedule*

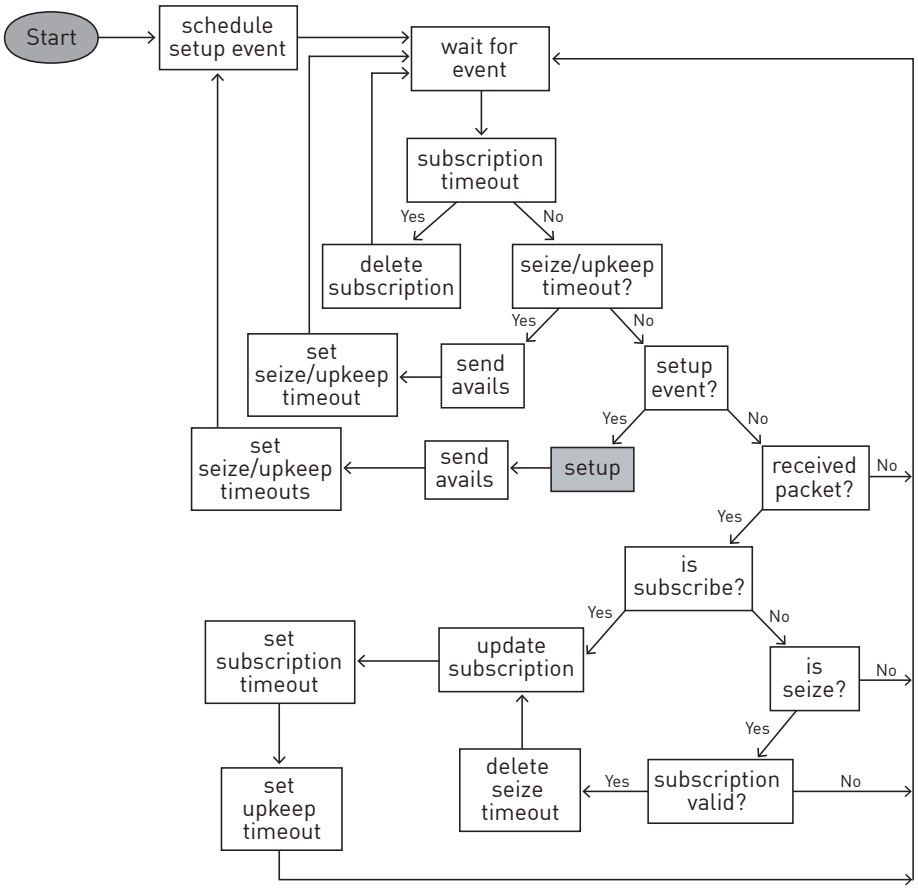

FIGURE 4.6: NSCA workflow for the network part.

setup event, where the timestamp for the next *setup* process is determined, followed by the state *wait for event*. The *subscription timeout* and the *seize/upkeep timeout* are identified and handled individually. In case of a *setup* event, the corresponding setup process is executed and the distribution of `avail` messages to the subscribed data sources is triggered. The frequent execution of the setup process enables an immediate reaction to network parameter changes. The specific frequency of this execution depends on multiple aspects, such as network size and available computational performance. This process contains all required mechanisms required to determine the available data rates for each flow as well as allocating resources in the network. In case a packet has been received, it is differentiated between `subscribe` and `seize` packets. A `subscribe` message either implies a subscription from a new or an existing source. Depending on this, a new subscription is created or an existing updated, as well as all corresponding timeouts, e.g. the timeout value to_{sub}, are updated. A `seize` message resets a presently existing timeout to_{seize} in case the subscription is still active.

The data source workflow is presented in figure 4.7. It is divided into two parts, the *data sending loop* and the *event handling loop*. After the initialization a `subscribe` message

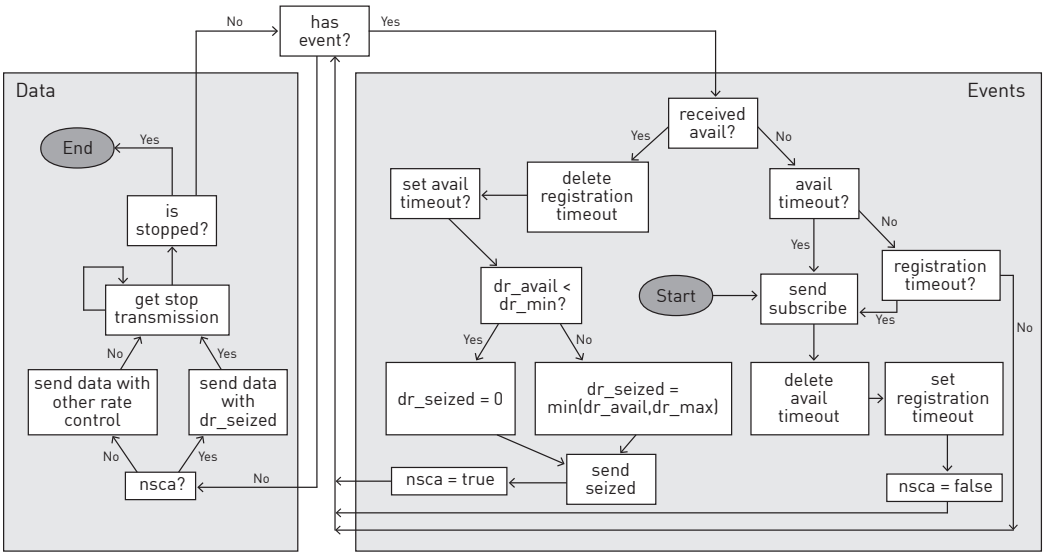

FIGURE 4.7: NSCA workflow for the source part.

is sent to initiate the communication with the network. The flag `nsca` is subsequently set to `false`. This flag represents the state whether *Network Supported Congestion Avoidance* is used or not. If `nsca==false` a traditional congestion control mechanism is required, such as *TCP Friendly Rate Control (TFRC)* [147].

At this point the source waits for events. In case no events are triggered, the *data sending loop* starts to emit data with a default congestion control scheme to avoid unnecessary latencies at the transmission start. The data sending loop is represented on the left side of figure 4.7. The data loop describes the current sending behavior. Data is either sent with the currently enabled rate control mechanism or via a default legacy end-to-end rate control. The loop terminates in case the transmission is stopped, or an event has been triggered that modifies the sending process. This is indicated by the state *get stop transmission*.

The *event handling loop* describes actions in case an `avail` or a `timeout` message has been received. An `avail` message contains the selected data rate dr_{seized}. *NSCA* is activated by setting `nsca==true` and the data loop is reentered in case no further events have to be processed. All `avail` messages from the network reset the currently cached dr_{seized} value. If either the `avail` or `registration` timeouts expire, a new subscription is sent to the network and the source continues to transmit data with a different rate control than NSCA. In case $dr_{avail} < dr_{min}$, dr_{seized} is set to zero to avoid misalignment. This case describes the situation where not sufficient resources are available in the network. The flexibility of the data stream is consequently critically limited. Once the

transmission has been terminated the subscription at the network entity automatically times out and no further interactions are required.

4.5.3 Network Rate Control Service

The network wide optimum stream distribution is calculated by the *Network Rate Control Service (NRCS)*. It represents an optimization approach inside the network to announce optimum data rates. The results are used to determine the lower and upper limits of stream data rate that are published to the data sources. An optimum transmission behavior assumes all present streams in the network to be NRCS-based. In case other congestion control mechanisms are used, the process cannot reliably determine the optimum data rate limits as each application individually modifies its sending behaviour without a global alignment. It is also assumed that all link capacities, including the maximum available data rate, are present to the NRCS. It serves as an initial approach and is part of the controller workflow illustrated in figure 4.6, represented by `setup`.

The NRCS can be designed in multiple ways, including a large set of different ingredients. It is possible to use linear programs, graph theory algorithms, or a trivial full-search approach to analyze the network environment and calculate an acceptable stream distribution. The used mechanism depends on the application scenario and must be precisely adjust to the given environment. In [R3] an example has been published that utilizes a modified graph tree search algorithm to implement the network rate control service.

4.6 Conclusion

This chapter presents general optimization approaches that can be used for network transmission tasks. It has been shown that *Linear Programming* and *Graph Theory* represent powerful approaches for multimedia content transport optimization. A discussion about metrics and norms presents different strategies to map multi-dimensional network characteristic vectors into a single real value. These mapping approaches enable the extension legacy algorithms and approaches to increase the benefit of their solutions when optimizing multimedia network traffic. As the default network infrastructure reflects the standards from the beginning of the Internet, it provides a coarse optimization granularity. Handling real-time multimedia traffic remains a challenge and must be addressed via other, partly revolutionary, approaches.

Besides general approaches of metric application and their design, the *Network Supported Congestion Avoidance (NSCA)* mechanism has been presented. It is seen as a

revolutionary and interactive application-network information exchange, that enables both communication components to explicitly announce details about the transmission parameters. The protocol itself and the corresponding workflows for network infrastructure and application are presented in detail. NSCA supports a multitude of different optimization approaches, depending on the present environment.

Chapter 5

Loss Domain Separation

Loss domain separation (LDS) refers to an approach that subdivides a network path into individual segments according to predefined segment characteristics. Thereby, it follows the *Performance Enhancing Proxies (PEP)* mechanisms described in [32]. Such characteristics can be the segment's delay, loss probability, or any other measurable parameter. It represents the basis for the performance of error correction mechanisms. For example, high packet loss and large segment delays require a stronger error correction. In case the network path is subdivided into multiple segments, each can be separately equipped with autonomous error correction schemes. Consequently, every scheme can be optimally adjusted to the underlying network segment. LDS focuses on a more efficient data transmission based on a suitable network path segmentation. The appropriate error correction scheme application allows a highly granular error correction process. Thereby, network segments are individually handled and provide direct information about their status. This is in contrast to a traditional end-to-end connection where information is merged from all network path segments and only available as an integrated unit.

The LDS approach applies to heterogeneous network environments with both, nearly error-free segments and segments with high error probability. A basic example is a default connection between an end-user and a video streaming server. The connection can be coarsely divided into a transport backbone network and the last mile at the end-users location. The backbone network is typically a wide area network with nearly error and congestion-free characteristics, but introduces large latencies. The end users network is typically a wireless LAN segment that introduces only small delays but significantly increases the probability of packet losses. Consequently, it is intuitive to split the connection at the node connecting the backbone network with the wireless in-home segment in order to create two *autonomous loss domains*.

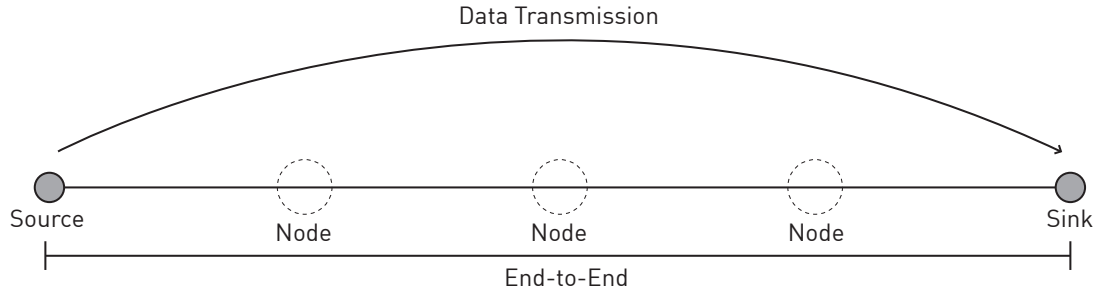

FIGURE 5.1: A traditional end-to-end transmission scheme.

Network connections frequently include mobile and other highly sensitive links in order to provide Internet access at any location. An error correction scheme is used to enable reliable transmissions. It either includes packet retransmissions (ARQ), prior added redundancy information (FEC), or a combination of both (HARQ) as presented in section 2.5.

The data source and sink are connected by network paths. These consist of multiple subsequent segments that are connected by nodes. Data is forwarded from one segment to another via these nodes and finally reaches the designated destination.

Error correction schemes can be used in two different ways, either on a *global connection level* or on a per *network segment* basis. The former is also known as end-to-end connection, whereas the latter describes a smaller part of the network path. This may include either a single or a combination of multiple physical links, but is ultimately treated as one virtual link. Figure 5.1 presents a traditional end-to-end transmission scheme, whereas figure 5.2 illustrates a schematic overview of a multi-hop transmission scheme.

In case an error correction scheme is applied on an individual segment, it is considered to represent this as an *atomic error correction unit*. Individual segment error protection provides a higher error coding precision and lowers the total network utilization. This protection mechanism only cares for a single segment and does not influence adjacent segments. Accordingly, if redundancy is required, it is only sent on this segment. The usage of atomic correction units breaks with the traditional end-to-end transmission principle [47] discussed in section 2.2 and shifts complexity from the network edge into the center.

Loss Domain Separation relies on multiple network segments on a path. These segments can be pure *physical* links, such as a direct switch-switch connection in a data center, or a *virtual* segment. A virtual segment is a logical conjunction of multiple segments, where each segment again can be physical or virtual. Figure 5.3 illustrates a virtual segment

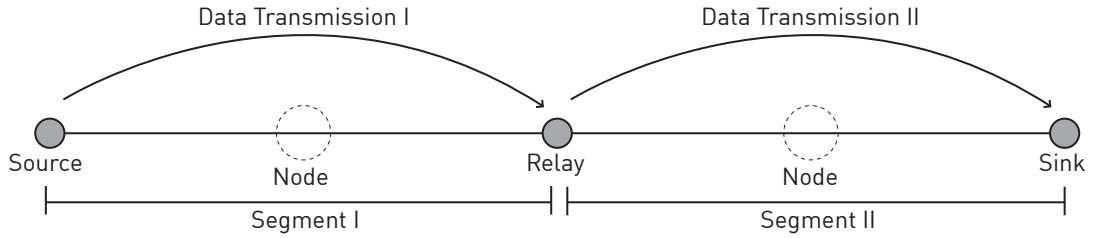

FIGURE 5.2: An LDS multi-hop transmission scheme.

in a network environment. In this figure, three physical segments are combined into one virtual segment. The original segment specific properties, such as latency and loss, are combined into a set of merged virtual properties. The end nodes do not recognize the different transmission domains as the path is considered to be consistent and continuous.

In this context LDS is applicable to both, the *data link* and the *networking* layer of the ISO/OSI model [38] as discussed in section 2.1. In case LDS is applied on higher abstraction layers, it additionally profits from the error correction mechanisms at the lower layer.

The overall transmission characteristics can be separated into two groups: *network-related* and *application-related*. The *network-related* characteristics are properties, that are directly originated by the network or its environment. They are assumed to be dynamic and severely vary over time due to changing stream occurrences and distributions. *Propagation delay*, *loss probability*, *jitter* and *throughput* are *network-related* characteristics. The *application-related* characteristics are introduced by the data application that initiates the transmission. They are assumed to be tunable and include properties as the maximum tolerated end-to-end *transmission latency* or the upper limit for the *residual error probability* at the data sink. Both remain constant over time. The application may adapt its sending parameters during the transmission, or only with new connections, depending on the current network status.

The LDS approach is motivated by the inhomogeneous network topology found in current wide area networks. Different types of links and varying physical layer conditions produce hardly predictable transmission conditions. As typical data sources and sinks face an end-to-end connection without any infrastructural details, they frequently use suboptimal transmission schemes, that consequently waste a significant share of network resources.

This heterogeneous structure leads to a *spatial dimension*, that provides a significant benefit for error coding schemes in future network environments. The *Loss Domain*

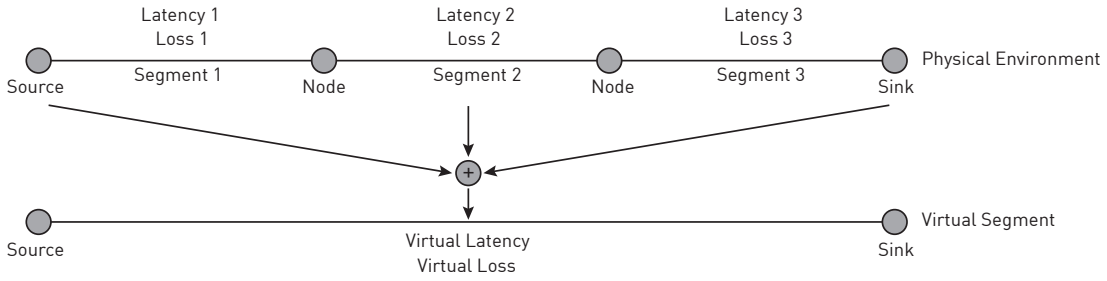

FIGURE 5.3: Virtual segment scheme.

Separation (LDS) principle is discussed considering present popular transport protocols. Basically, real-time audio and video content has been sent via RTP/UDP connections over network infrastructures. It unveils that also the *Transport Control Protocol (TCP)* [3] based solutions are adequate for streaming real-time content. Consequently, a standard for *Dynamic Adaptive Streaming over HTTP (DASH)* [27] has been emerged from the MPEG standard. It follows the ongoing paradigm shift towards an increased multimedia content population in global networks.

This chapter focuses on the theoretical details of *Loss Domain Separation (LDS)* for generic error correction schemes without time limitation. Subsequently, a packet retransmission scheme and a hybrid error correction, using ARQ and FEC, is discussed with respect to a strict time constraint.

5.1 Generic Error Correction Schemes

For the generic, non-time constraint error correction schemes, it is assumed that the communication is modeled as an *erasure channel* [52]. This channel classification has the characteristic that either a packet is perfectly received or it is completely lost. Single information flips are not possible. Erasure channels are commonly used to model IP network channels [52]. Network congestion is not considered in this scenario. It is also possible to indirectly reduce its impact by the presented techniques, as the total network load is reduced and the potential of congestion situations on individual network segments decreases, which leads to a lower number of required packets.

In this section, the focus is on a theoretical discussion relying on the *Noisy-Channel Coding Theorem* [35] as well as the application of a correction scheme with no time constraint.

The noisy channel coding theorem states that an arbitrary error coding technique is able to achieve an arbitrary low residual error probability in case it uses an infinite amount

of computation time [35]. The theoretical minimal required redundancy information RI_{min} for a segment with loss probability p is defined as shown in equation (5.1) [66].

$$RI_{min} = \frac{p}{1-p} \tag{5.1}$$

Equation (5.1) defines the theoretical minimum for the redundancy information in case all errors are completely corrected. If residual loss is allowed, the equation is modified as shown in equation (5.2) [66].

$$\begin{aligned} RI_{min,res} &= \frac{p - p_r}{1-p} \\ &= RI - \frac{p_r}{1-p} \end{aligned} \tag{5.2}$$

Equation (5.2) states that a large allowed residual loss leads to less redundancy information on the channel. It follows that the RI is linearly decreasing with p_r.

RI_{min} and $RI_{min,res}$ are valid for any error coding scheme in case no time constraints are present. General values for RI are consequently larger than RI_{min} as time is an inevitable and critical characteristic in productive environments. Accordingly, less time is spent for error correction and the higher the probability for error occurrences, the higher is the required redundancy information RI.

Error correction consequently increases the data rate due to the additional information RI. Let D_0 denote the original data rate of a source application and RI represents the required redundancy introduced by the error correction scheme. The ultimate data rate D emitted by the source in the network is

$$D = D_0 \cdot (1 + RI). \tag{5.3}$$

Equation (5.4) defines the *redundancy overhead* λ that relates the *coded* to the *uncoded* streaming data rate.

$$\begin{aligned} \lambda &= \frac{D}{D_0} \\ &= 1 + RI \quad \geq 1 \end{aligned} \tag{5.4}$$

λ includes data as well as redundancy and thus always attains values greater or equal 1. If $\lambda = 1$, all segments have zero probability of loss and no redundancy information is required. It follows $D = D_0$. In case $\lambda > 1$, the segment introduced errors and redundancy information is required and included in the transmission. As the error probability $p_i > 0$ for at least one segment i, it follows $D > D_0$.

In the following, the definition of λ is used to compare the data rates for *end-to-end* and *split* transmission scenarios with packet retransmissions.

The end-to-end transmission scenario is used as a reference model. It considers a *single* virtual segment representing the network-wide connection between source and sink. Assume that m subsequent segments build a network path. Each segment i has an individual loss probability $0 \leq p_i \leq 1$. Following [R10], the end-to-end loss probability p_{e2e} is

$$p_{e2e} = 1 - \prod_{i=1}^{m}(1 - p_i). \tag{5.5}$$

The average per link data rate D_{e2e} in an end-to-end transmission scheme with retransmissions is

$$\begin{aligned} D_{e2e} &= (1 + RI_{e2e}) \cdot D_0 \\ &= \left(1 + \frac{1 - \prod_{i=1}^{m}(1 - p_i)}{\prod_{i=1}^{m}(1 - p_i)}\right) \cdot D_0 \\ &= D_0 \cdot \underbrace{\frac{1}{\prod_{i=1}^{m}(1 - p_i)}}_{:=\lambda_{e2e}}. \end{aligned} \tag{5.6}$$

According to equation (5.4) the end-to-end redundancy overhead is achieved as

$$\lambda_{e2e} = \frac{1}{\prod_{i=1}^{m}(1 - p_i)}. \tag{5.7}$$

Splitting a transmission path yields m segments, each with a distinct error probability p_i, and thus represents the scenario with *Loss Domain Separation*. The average split data rate per link D_{split} is calculated as the sum over all segment data rates and divided by the number of segments. The derivation of D_{split} is shown in equation (5.8).

$$\begin{aligned} D_{split} &= \frac{\sum_{i=1}^{m}(1 + RI_i)}{m} \cdot D_0 = \frac{D_0}{m} \cdot \sum_{i=1}^{m}\left(\frac{1}{1 - p_i}\right) \\ &= \frac{D_0}{m} \cdot \frac{\sum_{i=1}^{m}\left[\prod_{j=1, j\neq i}^{m}(1 - p_i) \cdot (1 - p_j)\right]}{\prod_{i=1}^{m}(1 - p_i)} \\ &= D_0 \cdot \underbrace{\frac{\sum_{i=1}^{m}\left[\prod_{j=1, j\neq i}^{m}(1 - p_i) \cdot (1 - p_j)\right]}{m \cdot \prod_{i=1}^{m}(1 - p_i)}}_{=:\lambda_{split}} \end{aligned} \tag{5.8}$$

Analog to equation (5.4), the split transmission redundancy overhead is achieved as

$$\lambda_{split} = \frac{\sum_{i=1}^{m}\left[\prod_{j=1, j\neq i}^{m}(1 - p_i) \cdot (1 - p_j)\right]}{m \cdot \prod_{i=1}^{m}(1 - p_i)}. \tag{5.9}$$

A basic application scenario for a split error correction scheme includes only two subsequent network segments. Assume the case of a two segment path with error probabilities p_1 and p_2 for segment `I` and `II`, respectively. Let $0 < p_1 < 1$ be arbitrary, but fixed.

The loss probabilities can have two different relations:

- **Nearly equal:** This implies that both probabilities are approximately equal and thus $p_1 \approx p_2$.
- **Highly different:** In this case, one error probability significantly dominates the other and it holds $p_1 >> p_2$.

Due to the definitions of D_{e2e} and D_{split} in equations (5.6) and (5.8), it is sufficient to focus on the overhead factors λ_{e2e} and λ_{split} defined in equation (5.7) and (5.9).

An efficiency factor Λ is defined as shown in equation (5.10). It represents the advantage of a split approach compared to an end-to-end transmission approach considering the data rates.

$$\Lambda = \frac{\lambda_{split}}{\lambda_{e2e}} \tag{5.10}$$

The efficiency factor Λ can attain multiple values:

- $\Lambda = 1$: It implies that both approaches achieve the same efficiency.
- $\Lambda < 1$: It reflects that $\lambda_{split} < \lambda_{e2e}$ and thus the split approach is less efficient than the end-to-end transmission.
- $\Lambda > 1$: It reflects that $\lambda_{split} > \lambda_{e2e}$ and thus the split transmission approach efficiency outperforms the default end-to-end transmission scheme.

Consider the case where two segments have nearly equal error probabilities, that is $p_1 \approx p_2$. An end-to-end approach leads to the general form derived from equation (5.7) as shown in equation (5.11).

$$\begin{aligned} \lambda_{e2e}^{\approx} &= \frac{1}{(1-p_1)\cdot(1-p_2)} \\ &\approx \frac{1}{(1-p_1)^2} \\ &= \frac{1}{p_1^2 - 2\cdot p_1 + 1} \end{aligned} \tag{5.11}$$

Figure 5.4 shows the evaluation of $\lambda_{e2e}^{\approx}$ for the multiple values of p_1. Note that in this figure $\lambda_{split}^{\approx} == \lambda_{e2e}^{\neq}$ and both are consequently represented by only one line.

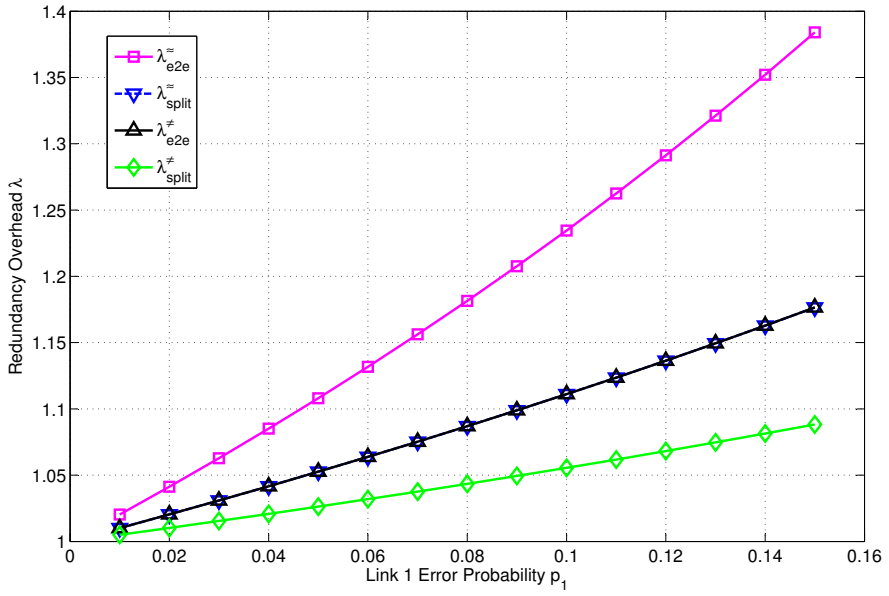

FIGURE 5.4: Evaluation of λ. Note that $\lambda^{\approx}_{split}$ equals $\lambda^{\neq}_{e2e}$.

The split transmission application leads to the redundancy overhead λ_{split} as shown in equation (5.12).

$$\begin{aligned}\lambda^{\approx}_{split} &= \frac{(1-p_1)+(1-p_2)}{2\cdot(1-p_1)\cdot(1-p_2)} \\ &\approx \frac{2-2\cdot p_1}{2\cdot(1-p_1)^2} \\ &= \frac{1}{1-p_1}\end{aligned} \tag{5.12}$$

Figure 5.4 also illustrates the evaluation of $\lambda^{\approx}_{split}$. The illustration shows that $\lambda_{e2e} \geq \lambda_{split}$ and identifies a significant reduction of the total required redundancy in case the split transmission approach is used. For example, if both segments have an error probability of $p_1 = p_2 = 0.07$, the total redundancy reduction is 0.09. In the case the application source data rate is $D_0 = 1\ Mbit/s$, the split approach implies an average reduction of 0.09 $Mbit/s$ per segment per transmission. The data rate reduction scales linearly with the number of transmissions on those segments.

The reduction factor $\Lambda^{\approx}$ for approximately equal loss probabilities is presented in equation (5.13).

$$\Lambda^{\approx} = \frac{\lambda_{split}}{\lambda_{e2e}} = \frac{(1-p_1)^2}{1-p_1} = 1-p_1 \tag{5.13}$$

The structure of $\Lambda^{\approx}$ unveils that a larger segment error probability leads to a higher data rate reduction in case of a split transmission approach.

Consider the scenario with one dominating error probability segment. For this case it is assumed that $p_1 >> p_2$. Equation (5.14) shows the corresponding redundancy efficiency for an end-to-end approach.

$$\begin{aligned} \lambda_{e2e}^{\neq} &= \frac{1}{(1-p_1)\cdot(1-p_2)} \\ &= \frac{1}{1-p_1-p_2+p_1\cdot p_2} \\ &\approx \frac{1}{1-p_1} \end{aligned} \tag{5.14}$$

Equation (5.15) shows the redundancy overhead for a split application with unequal segment error probabilities.

$$\begin{aligned} \lambda_{split}^{\neq} &= \frac{(1-p_1)+(1-p_2)}{2\cdot\prod_{i=1}^{2}(1-p_i)} \\ &= \frac{2-p_1-p_2}{2\cdot(1-p_1-p_2+p_1\cdot p_2)} \\ &\approx \frac{2-p_1}{2\cdot(1-p_1)} \\ &\approx \frac{1-\frac{p_1}{2}}{1-p_1} \end{aligned} \tag{5.15}$$

Figure 5.4 also illustrates the evaluation of $\lambda_{e2e}^{\neq}$ and $\lambda_{split}^{\neq}$ and it can be identified that $\lambda_{e2e} \geq \lambda_{split}$.

This is valid for non-time constraint error correction schemes, and consequently implies that a full-split transmission outperforms a default end-to-end transmission. It can be derived from equations (5.7) and (5.9) as follows:

$$\begin{aligned} & \lambda_{e2e} \geq \lambda_{split} \\ \Leftrightarrow \frac{1}{\prod_{i=1}^{m}(1-p_i)} &\geq \frac{\overbrace{\sum_{i=1}^{m}\left[\prod_{j=1,j\neq i}^{m}(1-p_i)\cdot(1-p_j)\right]}^{p'}}{m\cdot\prod_{i=1}^{m}(1-p_i)} \\ &= \frac{p'}{m\cdot\prod_{i=1}^{m}(1-p_i)} \\ &= \frac{1}{\underbrace{\frac{m}{p'}}_{M}\cdot\prod_{i=1}^{m}(1-p_i)} \\ &= \frac{1}{M\cdot\prod_{i=1}^{m}(1-p_i)} \end{aligned} \tag{5.16}$$

Since $\forall i : 0 \leq p_i \leq 1$ it follows $p' \leq m$. Consequently, $M = \frac{m}{p'} \geq 1$ and it holds $\lambda_{e2e} \geq \lambda_{split}$.

With the exemplary assumptions $p_1 = 0.07 >> p_2$ and $D_0 = 1\ Mbit/s$ the split transmission causes a reduction of approximately $0.03\ Mbit/s$ per link on average. This data rate reduction scales linearly with the number of transmissions.

In analogy to equation (5.13), the data rate reduction factor for a two link scenario with significant unequal error probabilities is

$$\Lambda^{\neq} = \frac{\lambda_{split}}{\lambda_{e2e}} = \frac{(1-p_1)\cdot(1-\frac{p_1}{2})}{1-p_1} = 1 - \frac{p_1}{2}. \tag{5.17}$$

For the case that both segments have approximately the same error probability, the application of a split error correction approach causes a data rate reduction of a factor $1-p$ per link on average, whereas if $p_1 >> p_2$ the data rate reduction per link on average is $1 - \frac{p_1}{2}$.

5.2 Time Constraint Automatic Repeat Request

This section describes the theoretical basis for time constraint packet retransmission schemes in multi-hop environments that respect the *predictable reliability under predictable delay (PRPD)* transmission principle [36, 66]. Whereas the original PRPD work proposes a complex adaptive hybrid error coding scheme for end-to-end scenarios, this section focuses on the underlying mathematics of ARQ correction schemes in multi-hop networks. As the transmission time is restricted, it is required to distribute the available end-to-end delivery time among all multi-hop network segments. This decreases the network traffic and finally leads to lower network traffic costs or to higher backup resource capacities in networks.

The globally predominant IP error correction approach is *Automatic Repeat Request (ARQ)*, as it is used in the popular layer 4 transport protocol TCP [3]. Each retransmission requires at least one full round trip time for signaling and resending data. If a delivery time limit is given, it is important to efficiently distribute the retransmission rounds, and therefore implicitly the coding time, among the segments. The end-to-end approach blurs the underlying network topology and the corresponding characteristics to a coarse virtual transmission link. If an error correction is applied to this virtual link, it adapts to its merged properties, e.g. round trip time and packet loss rate. If the error-prone links are handled individually, unaffected links are released from undesired traffic and the total network load is lowered. In case of time constraint transmissions, a

suitable coding time assignment is required to obtain the optimum, that is a *minimum* network load.

The ARQ scheme has a constant maximum delivery time per data packet. In case this time limit is not sufficient to deliver either the data packet or a redundancy packet to the receiver, the packet is considered as lost and thus skipped. ARQ maintains multiple retransmission rounds, each providing a specific number of retransmission packets. For time constraint transmissions, the number of ARQ rounds, and thus the required time, must be limited. Assume that the time consumption within one retransmission round is negligible, that means sending one or more packets requires always the same amount of time. Let $\vec{X} = [x_1, x_2, \ldots, x_m]$ denote the retransmission vector for an arbitrary ARQ scheme. Each element x_i corresponds to a specific number of packets to be sent in round i. Consequently, $m = |\vec{X}|$ represents the total number of elements $x_i \in \vec{X}$. The length of $\vec{X}$ always depends on the used retransmission scheme. Since each x_i reflects exactly one retransmission round, the total time consumption T of this scheme is $T = m \cdot RTT$, where RTT is the round trip time of the underlying communication channel.

In time-constraint transmission scenarios with residual loss requirements, m must be limited to avoid an exceeding of the maximum allowed transmission time Δ. The retransmission vector $\vec{X}$ must be properly limited. It can be limited by the total number of retransmission packets N_p. The number of required retransmission packets to reach the requested residual error-rate p_r on a channel with error probability p is calculated as shown in equation (5.18).

$$\begin{aligned} p_r &\geq p^{N_p+1} \\ \Leftrightarrow N_p &= \left\lceil \frac{\log(p_r)}{\log(p)} \right\rceil - 1 \end{aligned} \tag{5.18}$$

m can also be limited by the total number of possible retransmission rounds N_r. The number of rounds is selected in a way, that the given delivery time limit Δ minus the amount of a half RTT for the original packet transmission is not exceeded:

$$\begin{aligned} \Delta - \frac{1}{2} \cdot RTT &\geq N_r \cdot RTT \\ \Leftrightarrow N_r &= \left\lfloor \frac{\Delta}{RTT} - \frac{1}{2} \right\rfloor \end{aligned} \tag{5.19}$$

The size m of the vector is determined by the minimum of both and shown in equation (5.20).

$$m = \min(N_p, N_r) \tag{5.20}$$

In case $N_r \geq N_p$, there is an adequate amount of time to serve each packet with a separate round, thus $m = N_p$. For example, if $N_r = 6$ and $N_p = 4$ the corresponding retransmission vector would be $\vec{X} = [1, 1, 1, 1]$.

Otherwise $k = N_p - N_r$ packets must be additionally sent in an accumulated fashion during one round. Due to the i.i.d. erasure channel, accumulating the k packets in the last possible round m, e.g. $x_m = 1 + k$ reduces the expected number of redundancy packets. For the case that $N_r = 4$ and $N_p = 6$ there are $k = 2$ additional packets required in the last round, and the retransmission vector would be $\vec{X} = [1, 1, 1, 3]$. Both cases can be unified by defining the number of additional packets in the last round as $k = \max(0, N_p - N_r)$.

The required redundancy information RI_{ARQ} for a time limited ARQ scheme can be defined as presented in equation (5.21).

$$RI_{ARQ} = \sum_{i=1}^{N_r-1} p^i + (1+k) \cdot p^{N_r} \quad = \sum_{i=1}^{N_p-k-1} p^i + (1+k) \cdot p^{N_p-k} \tag{5.21}$$

Equation (5.22) defines η as the ratio between the theoretical optimum RI_{min} and the ARQ redundancy RI_{ARQ}. It represents a generic redundancy efficiency metric with values $0 \leq \eta \leq 1$.

$$\eta = \frac{RI_{min,res}}{RI_{ARQ}} \tag{5.22}$$

RI_{min} does not depend on any specific error correction approach, but is based on the channel coding theorem [35].

With equations (5.2), (5.22) and (5.21), it follows:

$$\begin{aligned} \eta &= \frac{RI_{min,res}}{RI_{ARQ}} \\ &= \frac{\frac{p-p_{res}}{1-p}}{\sum_{i=0}^{N_p-k-1} p^i - 1 + (1+k) \cdot p^{N_p-k}} \\ &= \frac{\frac{p-p_{res}}{1-p}}{\frac{1-p^{N_p-k}}{1-p} - 1 + (1+k) \cdot p^{N_p-k}} \\ &= \frac{p - p_{res}}{p - p^{N_p-k} + (1+k) \cdot p^{N_p-k} - (1+k) \cdot p^{N_p-k+1}} \end{aligned} \tag{5.23}$$

This efficiency measurement depends on the assigned time due to N_r in the definition of k. Figure 5.5 illustrates the continuous efficiency η of a single link with fixed round trip time of 10 ms, a fixed residual loss rate of 10^{-6}, and three different error probabilities. It can be seen that with an increasing error probability the slope of the efficiency decreases and thus more time is required to reach the same efficiency values compared to a link

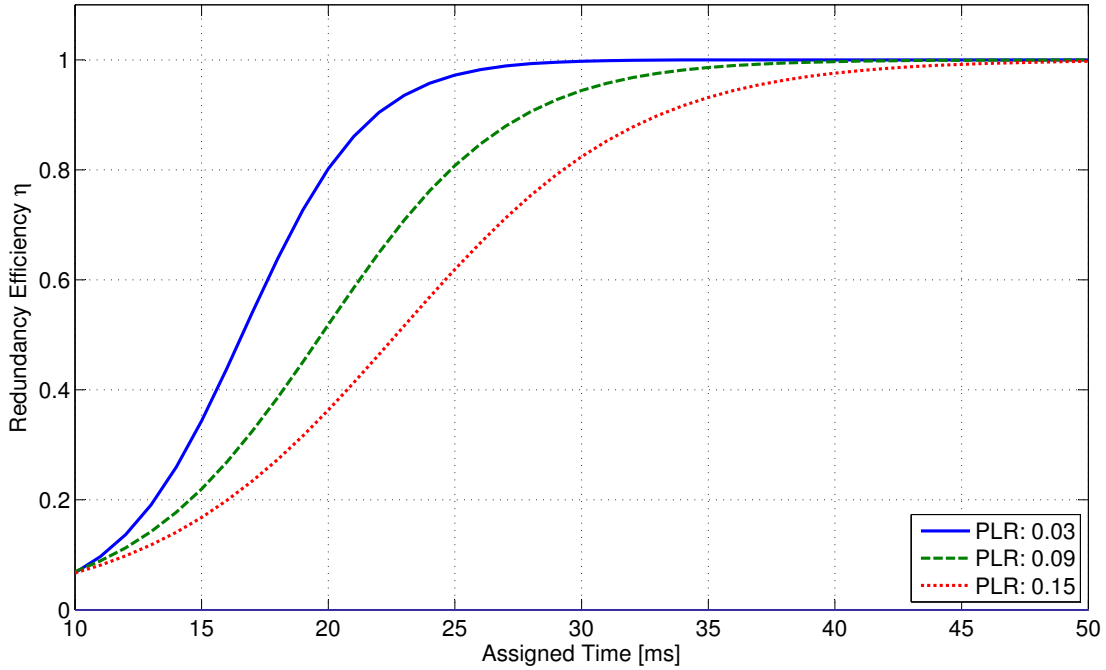

FIGURE 5.5: Continuous redundancy efficiency for time-constraint ARQ with $RTT = 10\ ms$ and $p_{res} = 10^{-6}$.

with lower error probability. In case of a non-continuous round numbers, the efficiency function would have a step shape. It can be seen, that η approaches 1 in case more time is spent on the retransmission scheme, and thus $RI_{ARQ} \rightarrow RI_{min}$.

With end-to-end transmissions, the network between source and sink is considered as a virtual link that incorporates the characteristics of the underlying real network parts as blurred information. Error coding application would affect the whole path at once without any differentiation between the individual network segments. The distribution is trivial by spending the total time budget Δ on this single virtual link. With multi-hop transmissions the network path between source and sink is considered to be a set of individual network segments, each with its specific characteristics.

A *time budget distribution problem (TBDP)* arises in case data must be sent over multiple individual network segments within a specific time period. Basically, the delivery time is based on the underlying network round trip delay. ARQ retransmission takes at least one full round trip time. For a given time budget Δ the challenge arises how to distribute the available delivery time, thus retransmission rounds, efficiently amongst the network and its individual segments.

A basic full-search approach assumes n segments representing a network path. Each segment i is described by its round trip time RTT_i and packet loss rate p_i, with $1 \leq i \leq n$. Let the total end-to-end target delay be Δ and let p_{res} reflect the end-to-end residual error rate. Assume an equal distribution $p_{res,i}$ of p_{res} among all links. Let

	L_1	L_2	L_3	L_4	L_5	L_6	L_7
PLR	0.07	0.06	0.05	0.03	0.02	0.02	0.02
RTT	10 ms	8 ms	17 ms	12 ms	11 ms	19 ms	6 ms

TABLE 5.1: Network path example for time-constraint ARQ example.

$X = [x_1, x_2, \ldots, x_n]$ be the vector that contains the number of required retransmission packets for each link $1 \leq i \leq n$. The total search space $\mathbb{X} = (X_1, X_2, \ldots, X_q)$ of all possible combinations is defined as shown in equation (5.24).

$$\begin{aligned} \forall\ 1 \leq j \leq |\mathbb{X}| : \qquad & \textstyle\sum_{i=1}^{n} x_{j,i} \cdot RTT_i \ \leq \Delta - \tfrac{1}{2} \cdot \textstyle\sum_{i=1}^{n} RTT_i \\ \text{subject to} \quad 1 \leq x_{j,i} \leq \left\lceil \tfrac{log(p_{res,i})}{log(p_i)} \right\rceil - 1 \ &= N_p(i) \\ p_{res,i} \ &= 1 - \sqrt[n]{(1 - p_{res})} \end{aligned} \tag{5.24}$$

$x_{j,i}$ represents the number of retransmissions on link i for the combination X_j. $N_p(i)$ reflects the total number of required packets on link i. The size q of $\mathbb{X}$ is upper bounded by $q \leq |\mathbb{X}| = \prod_{i=1}^{n} N_p(i)$. The full-search evaluates all vectors $X_j \in \mathbb{X}$.

Therefore, the corresponding retransmission vectors RP must be created. Since the error probability in an i.i.d. channel decreases exponentially with the number of rounds, it is sufficient to store all accumulation packets $k_{j,i} = N_p(i) - x_{j,i}$ in the last round, as shown in equation (5.21). Thus, for the elements $rp_{j,i}$ of the retransmission vector RP holds:

$$\forall 1 \leq j \leq q : \forall 1 \leq i \leq n : rp_{j,i} = [v_1, v_2, \ldots, v_{x_{j,i}} + k_{j,i}] \tag{5.25}$$

with $v_1 = v_2 = \ldots = v_{x_{j,i}} = 1$ representing the number of packets in the corresponding round. The resulting average redundancy information RI_j can be calculated as shown in equation (5.26).

$$\begin{aligned} \bar{R}I_j &= \frac{\sum_{i=1}^{n} \sum_{w=1}^{x_{j,i}} p_i^w \cdot v_w}{n} \\ &= \frac{1}{n} \cdot \sum_{i=1}^{n} \left[\sum_{w=1}^{x_{j,i}-1} p_i^w \cdot v_w + p_i^{x_{j,i}} \cdot \left(v_{x_{j,i}} + k_{j,i} \right) \right] \end{aligned} \tag{5.26}$$

The global minimum redundancy $\bar{R}I_{min}$ value among all RI_j is

$$\bar{R}I_{min} = \min_{\mathbb{X}} (\bar{R}I_j). \tag{5.27}$$

For the evaluation, a seven-segment path is assumed, where each link is tagged with a specific round-trip time and packet loss rate value. According to [122], it is supposed

	Time	RP	RI_{ARQ}	RI_{min}
L_1	35.0 ms	$\begin{bmatrix}1 & 1 & 3\end{bmatrix}$	0.0759	0.0753
L_2	28.0 ms	$\begin{bmatrix}1 & 1 & 2\end{bmatrix}$	0.0640	0.0638
L_3	59.5 ms	$\begin{bmatrix}1 & 1 & 2\end{bmatrix}$	0.0528	0.0526
L_4	30.0 ms	$\begin{bmatrix}1 & 2\end{bmatrix}$	0.0318	0.0309
L_5	27.5 ms	$\begin{bmatrix}1 & 2\end{bmatrix}$	0.0208	0.0204
L_6	47.5 ms	$\begin{bmatrix}1 & 2\end{bmatrix}$	0.0208	0.0204
L_7	21.0 ms	$\begin{bmatrix}1 & 1 & 1\end{bmatrix}$	0.0204	0.0204
$E2E$	207.5 ms	$\begin{bmatrix}1 & 8\end{bmatrix}$	0.7095	0.3189

TABLE 5.2: Time-constraint ARQ example evaluation.

that the residual error-rate for each link is $p_{res,i} = 10^{-6}$, thus for the end-to-end case it holds $p_{res} = 1 - (1 - p_{res,i})^n$. The total allowed delivery time is set to $\Delta = 250\ ms$. Table 5.1 presents the links and their corresponding characteristics. In this example, without loss of generality, the links on the path are ordered in a packet loss rate (PLR) descending manner. In table 5.2 it can be seen that the average redundancy values per link nearly approximate the individual theoretical lower bounds of each link. The average real redundancy per link is

$$\bar{RI}^{mh}_{ARQ} = \frac{\sum_{i=1}^{7} RI_{ARQ}(L_i)}{7} = 0.040931, \tag{5.28}$$

whereas the theoretical average per link is

$$\bar{RI}^{mh}_{min} = \frac{\sum_{i=1}^{7} RI_{min}(L_i)}{7} = \frac{\sum_{i=1}^{7} \frac{p_i - p_{res}}{1-p_i}}{7} = 0.040555. \tag{5.29}$$

The result is an end-to-end repetition vector $RP = \begin{bmatrix}1 & 8\end{bmatrix}$, that generates redundancy per link of $RI^{e2e}_{ARQ} = \bar{RI}^{e2e}_{ARQ} = 0.7095$.
The repetition vectors of the links have different lengths, according to their PLR and RTT values. Consequently links with high packet loss probabilities have more time to perform more rounds than better links. Therefore, the resulting redundancy is lowered. The application of the proposed algorithm achieves a redundancy performance on the multi-hop path scenario that is only $3.76 \cdot 10^{-4}$ below the theoretical limit.

5.3 Hybrid Automatic Repeat Request

Hybrid Automatic Repeat Request (HARQ) or *Hybrid Error Correction (HEC) schemes* include both, *Automatic Repeat Request (ARQ)* [34] and *Forward Error Correction (FEC)* [54] techniques, as introduced in section 2.5. A hybrid scheme represents a composition of two individual schemes, and consequently its characteristics are more

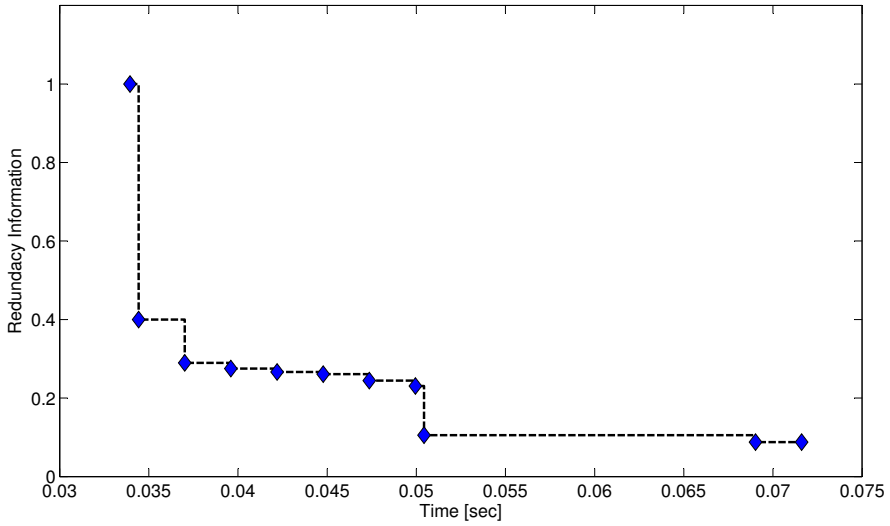

FIGURE 5.6: Redundancy Information (RI) shape of an Ω table [R9].

complex and incorporate a multitude of adjustable parameters. The time behavior of these schemes depends on a combination of parameters, including the retransmission rounds, the code block size, and the code rate. These ingredients significantly influence the overall performance. Deriving a linear and approximately monotonic relation between latency and redundancy can be achieved with a single error correction scheme. The combination of both leads to a step-shaped characteristic as illustrated in figure 5.6. In this case, local extreme points can exist due to a variation of ARQ rounds and FEC block sizes, depending on the environmental characteristics. The application of a hybrid error correction scheme in combination with *Loss Domain Separation (LDS)* is a promising optimization task for future media Internet scenarios. This is especially important for transmission environments with time constraints.

In this section the application of LDS based on a hybrid scheme and the *HARQ Saturation Problem* is presented and discussed. The saturation problem represents the case when a network is radically segmented for a time constraint transmission while using hybrid error correction approaches. The following considerations are based on the *Adaptive Hybrid Error Correction (AHEC)* framework.

5.3.1 Adaptive Hybrid Error Correction Framework

The *Adaptive Hybrid Error Correction (AHEC)* framework constitutes the mathematical foundation of the HARQ Saturation problem, that is discussed in the next section. This framework has been developed in [36] at Saarland University, and is illustrated in figure 5.7. It utilizes statistical calculations based on the *Gilbert-Elliot channel model* [66] to adapt the parameters of a hybrid error correction scheme to the underlying channel.

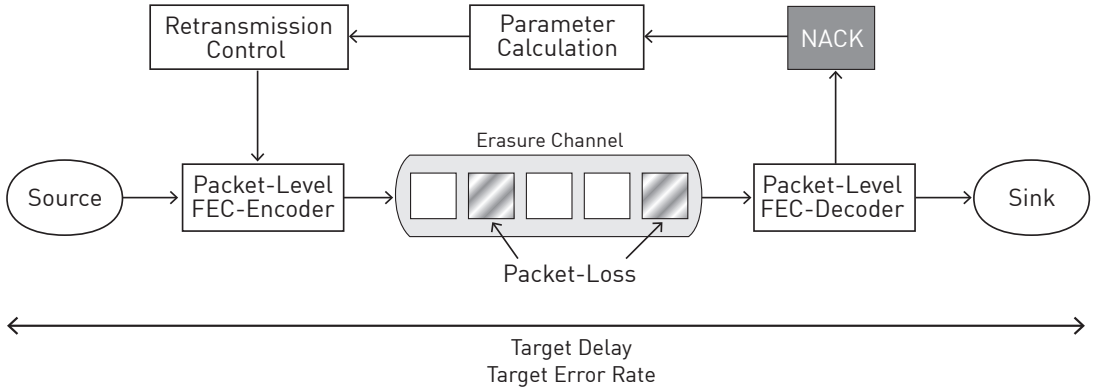

FIGURE 5.7: The Adaptive Hybrid Error Correction (AHEC) Framework.

AHEC applies a systematic block code with the special characteristic that the information content is sent first without a directly following redundancy block. The block code's redundancy is sent in case the receiver collects too little information owing to transmission distortions, and it is not able to process the remaining available information. It consequently sends a *negative acknowledgement (NACK)* to the sender and orders redundancy for the present information block.

In particular, the AHEC framework differs from default hybrid error correction schemes as it *guarantees* a maximum transmission time Δ between source and sink while ensuring an upper bound for the residual error rate P_{target}. To achieve this, it exploits a set of *system* and *coding* parameters. The system parameters are set up by the present network environment and its status. It mainly relies on the following variables:

- **Channel State Information:** It includes the transition probabilities α and β for each receiver and multicast group. It directly influences the Gilbert-Elliot channel model.
- **Round Trip Time (RTT):** The *RTT* reflects the current basic latency between the source and sink in the network.
- **Average Packet Interval:** This *time* measure T_s represents the average time required to send a single packet. This measure depends on the link data rate and the packet size.
- **Number of Receivers:** The *number of receivers* N_{recv} directly influences the complexity and the search space. The more receivers are available, the more complex are the calculations, as the transmission must be optimized to satisfy every receiver's requirements.

The *Channel State Information* ($\vec{CSI}$) is calculated according to the NACK population based on a Maximum-Likelihood Estimation (MLE) [36]. The remaining system parameters can be potentially evaluated with the help of RTCP messages.

The coding parameters for the AHEC framework are:

- the coding block length K,
- the number of redundancy packets sent ahead with the original transmission N_{pkt},
- the number of total retransmission rounds N_r, and
- the number of individual packet retransmissions per round $\vec{X}$.

The coding parameters are crucial for the adherence of Δ and P_{target}. For example, the number of retransmissions rounds N_r depends on the available round trip time and influences the error correction characteristics and performance.

The AHEC framework eventually states and solves the following optimization problem [148] as shown in equations (5.30) and (5.31).

$$RI_{min} = \text{argmin}\left(RI\left(K\left(N_r\right), N_{pkt}, \vec{X}, N_{recv}, \vec{CSI}\right)\right) \tag{5.30}$$

$$\begin{aligned} K \cdot T_s + \left(\frac{1}{2} + N_r\right) \cdot RTT &\leq \Delta \\ PLR\left(K\left(N_r\right), N_{pkt}, \vec{X}, N_{recv}, \vec{CSI}\right) &\leq P_{target} \end{aligned} \tag{5.31}$$

In section 3.5, the *Predictably Reliable Real-Time Transport (PRRT)* [66] has been presented. PRRT represents an online available[1] implementation of the AHEC framework extended by mechanisms for the professional use in IP network environments.

5.3.2 HARQ Saturation Problem

The description of the *HARQ Saturation Problem* [R9] requires a set of configurations ω. A single configuration ω represents the relation between the input parameters of a hybrid error correction scheme and the corresponding redundancy information RI, as well as network related parameters, such as round-trip time. Here, the inputs are limited to latency δ, residual error rate p_r, and the abstract network parameter N. Generally, it is possible to extend the description of ω by further parameters and adapt the required operations and algorithms. An exemplary configuration ω is presented in equation (5.32). It shows a quadruplet, where the first two entries represent the

[1] http://www.nt.uni-saarland.de/en/projects/running-projects/prrt.html (2015/01/29)

configuration parameters, the third entry reflects the generated RI when using these parameters, and the last entry N represents any other network specific parameters.

$$\omega = (\delta, p_r, RI, N) \tag{5.32}$$

Multiple configurations are grouped in a configuration table Ω with a specific upper limit for latency δ_{max} and error-rate $p_{r,max}$. An Ω table is consequently defined as shown in equation (5.33).

$$\Omega = \{\omega \mid 0 \leq \delta \leq \delta_{max} \cup 0 \leq p_r \leq p_{r,max}\} \tag{5.33}$$

Ω is a discrete set and contains only valid configurations ω that do not exceed the delay and residual loss thresholds. It is assumed that a configuration $\omega \in \Omega$ is generated by the aforementioned AHEC framework [36]. The granularity of the table directly corresponds with the inputs of the AHEC framework. The number of entries ω is related to the AHEC setup by variables as delay and residual error step size, as AHEC iteratively generates configurations ω with discrete delay and residual loss values. It is also possible to compute the required configurations including the RI online, but due to the high computational complexity, it was proposed to use precalculated Ω tables.

An exemplary illustration of the total redundancy shape of an Ω table with hybrid error correction configurations is given in figure 5.6. In case only a small amount of time is available, a hybrid scheme is configured to use FEC. The application of ARQ in this case is highly inflexible as each retransmission round consumes at least one full RTT. If more time is available, ARQ should be preferred due to its reactive correction characteristic. It avoids unnecessary redundancy information in the network. All cases in-between lead to a combinatorial optimization task to determine the right balance between the utilization of FEC and ARQ.

In the following, a precalculated Ω table represents an atomic unit per network segment that contains a set of individual segment configurations. Environments with end-to-end error correction schemes solely face this optimization challenge with one Ω table. As a multi-hop transmission path is split into multiple segments, the problem extends to consider a multitude of individual Ω tables.

Let Ω_i be the corresponding table associated with the network segment $i \in \{1, \ldots, n\}$. Multimedia transmissions have strict time and loss constraints, thus define a maximum tolerable upper end-to-end limit for both. Assume Δ and P_r to be the maximum allowed

transmission latency and residual error-rate, respectively. Let t_i be the time portion from Δ assigned to segment i. It must subsequently hold:

$$\sum_{i=1}^{n} t_i \leq \Delta. \tag{5.34}$$

Let $p_{r,i}$ be the amount of residual loss assigned to segment i. It holds for the residual error rate P_r

$$1 - \prod_{i=1}^{n}(1 - p_{r,i}) \leq P_r. \tag{5.35}$$

For the following discussion, only the time distribution is considered since the focus is on multimedia content as described in chapter 3. This type of content is able to tolerate a specific amount of residual loss. Late packets imply a playback delay that significantly decreases the *Quality of Experience (QoE)* and *Quality of Service (QoS)* at the user's side. Variations in segment RTTs are more likely than residual loss variations. The input parameter for the residual loss $p_{r,i}$ on a segment i with loss probability p_i generally influences the RI outcome of an error correction scheme according to equation (5.2). The residual error probability for a transmission is typically significant lower than the path error probability. In table 3.1, the ITU Y.1541 [121] recommendation for the QoS service classes 6/7 states a maximum packet loss rate of 10^{-5}. Assuming an average link error probability in the order of 10^{-3}, the amount of saved RI is ${}^{10^{-5}}/_{(1-10^{-3})} = 1.001 \cdot 10^{-5}$. The residual error rate $p_{r,i}$ on a segment i must not exceed the link error probability p_i, that is $p_{r,i} \leq p_i$. It additionally must satisfy the end-to-end residual loss requirement. For practical approaches, the value of $p_{r,i}$ should scale with the p_i. In case an error-prone link gets a larger value of allowed residual loss, the applied error correction scheme can introduce moderate amounts of RI as occurring errors must not be perfectly corrected. An MCKP-based distribution approach for multiple constraints, such as delay and residual error rate, is presented in section 6.4 and [R8].

The following *saturation point* discussion focuses on the distribution of coding delay only. According to equation (5.35), the residual loss values $p_{r,i}$ are assumed to be evenly distributed on the network path as shown in equation (5.36).

$$\forall i: \; p_{r,i} = \sqrt[n]{1 - P_r} \tag{5.36}$$

As the entries $\omega \in \Omega$ are sampled discretely, for each Ω table a *minimum delay* configuration entry ω_{min} exists. Selecting the ω_{min} from all available Ω tables on the path generally represents a valid distribution, as the time constraint given by equation (5.34) is not violated.

The minimum required time portion Δ^- for the total network path is shown in equation (5.37).

$$\begin{aligned}\Delta^- &= \sum_{i=1}^{n} \delta_{i,min} \\ &= \sum_{i=1}^{n} \min\left(\delta | \omega \in \Omega_i\right)\end{aligned} \tag{5.37}$$

Following the channel coding theorem [35] this distribution generates a maximum of redundancy information and is consequently considered as not suitable. Δ^- represents the lower latency bound for error correction time distribution as indicated in equation (5.38). It states the absolute minimum of time required to spend across the segments on the path.

$$\Delta^- \leq \sum_{i=1}^{n} t_i \leq \Delta \tag{5.38}$$

With equations (5.37) and (5.38), it is possible to define a range of valid time assignments $[\delta_{i,min}, \ldots, R_i]$ for a segment i.

Consider γ_i as the amount of available time on segment i in case the minimum delay configuration $\omega_{min,i}$ has already been assigned. For an adequate time distribution it must hold:

$$\begin{aligned}\Delta &\geq \sum_{i=1}^{n} R_i \\ &= \sum_{i=1}^{n} \left(\delta_{i,min} + \gamma_i\right).\end{aligned} \tag{5.39}$$

For a network path comprising of S segments, this leads to the overall available time $\Gamma(S)$ that can be distributed across the path as shown in equation (5.40).

$$\begin{aligned}\Gamma(S) &= \Delta - \sum_{i=1}^{S} \delta_{i,min} \\ &= \sum_{i=1}^{S} R_i - \sum_{i=1}^{S} \delta_{i,min} \\ &= \sum_{i=1}^{S} \left(R_i - \delta_{i,min}\right) \\ &= \sum_{i=1}^{S} \gamma_i\end{aligned} \tag{5.40}$$

Combining equation (5.38) and (5.40) leads to:

$$\delta_{i,min} \leq t_i \leq \delta_{i,min} + \gamma_i = R_i. \tag{5.41}$$

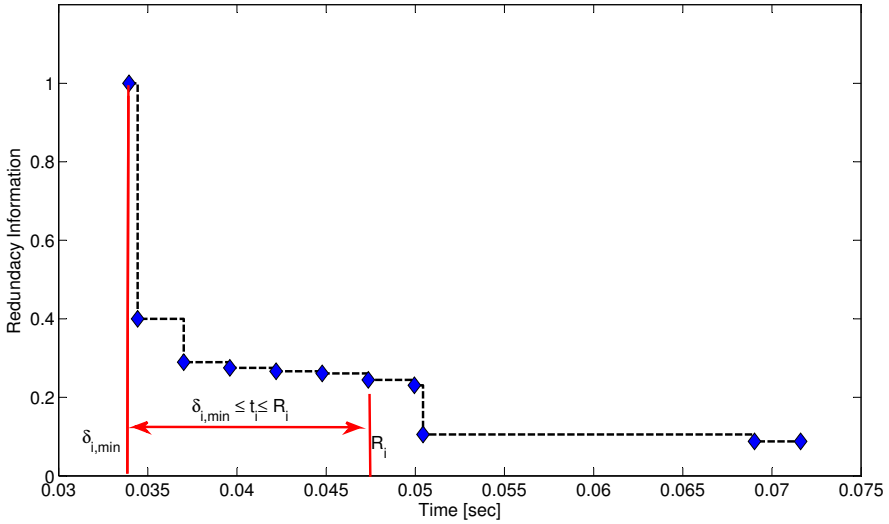

FIGURE 5.8: Time influence in Ω redundancy information table [R9].

A smaller value of R_i leads to a lower number of possible configuration entries for segment i as the available time decreases. The error correction performance is significantly lowered as shown in figure 5.8. It is not suitable to assign a large amount of time to a single segment only, as this segment would significantly dominate the others. The global time budget is severely reduced and the remaining segments are bound to a small amount of time. Figure 5.9 illustrates this fact for a path with two segments. Whereas the available time on one segment is increased from $R_1 \rightarrow R_1'$, the time on the second segment decreases accordingly from $R_2 \rightarrow R_2'$.

For the identification of the *saturation point* assume an evenly distributed time reserve γ_i over S segments of a path according to equation (5.42).

$$\gamma_i = \frac{\Delta - \sum_{i=1}^{S} \delta_{i,min}}{S} = \frac{\Gamma(S)}{S} \tag{5.42}$$

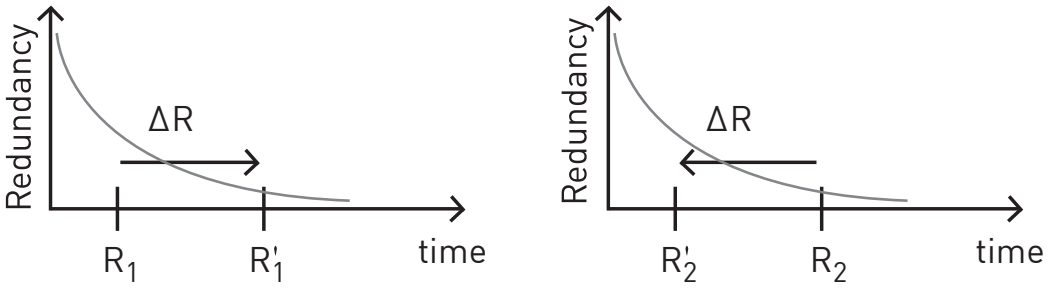

FIGURE 5.9: Time and redundancy interdependency for a two link scenario.

With equations (5.40) and (5.42) it follows:

$$\begin{aligned}
\Gamma(S) = \sum_{i=1}^{S} \gamma_i &= \Delta - \sum_{i=1}^{S} (\delta_{i,min}) \\
&= \sum_{i=1}^{S} (R_i - \delta_{i,min}) \\
&\geq \Delta - \sum_{i=1}^{S+1} \delta_{i,min} \\
&= \sum_{i=1}^{S+1} \left(R'_i - \delta_{i,min}\right) \\
&= \sum_{i=1}^{S+1} \gamma'_i \\
&= \Gamma(S+1).
\end{aligned} \tag{5.43}$$

It can be shown that the available time decreases with a larger number of segments:

$$\gamma_i = \frac{\Gamma(S)}{S} > \frac{\Gamma(S+1)}{S} > \frac{\Gamma(S+1)}{S+1} = \gamma'_i. \tag{5.44}$$

The required redundancy information consequently increases. This basic approach of simply increasing the number of segments on a network path does not tackle the problem of optimizing the transport of multimedia data. The application of LDS with an uncoordinated number of individual network segments decreases the overall transmission performance with hybrid error correction schemes. Equation (5.45) shows the existence of the HARQ segmentation saturation point S_0 that represents a spatial optimization bound.

$$\exists\ S_0 \in [1 \ldots S] :\ \forall s > S_0 :\ \sum_{i=1}^{S_0} RI(t_i) < \sum_{i=1}^{s} RI(t_i) \tag{5.45}$$

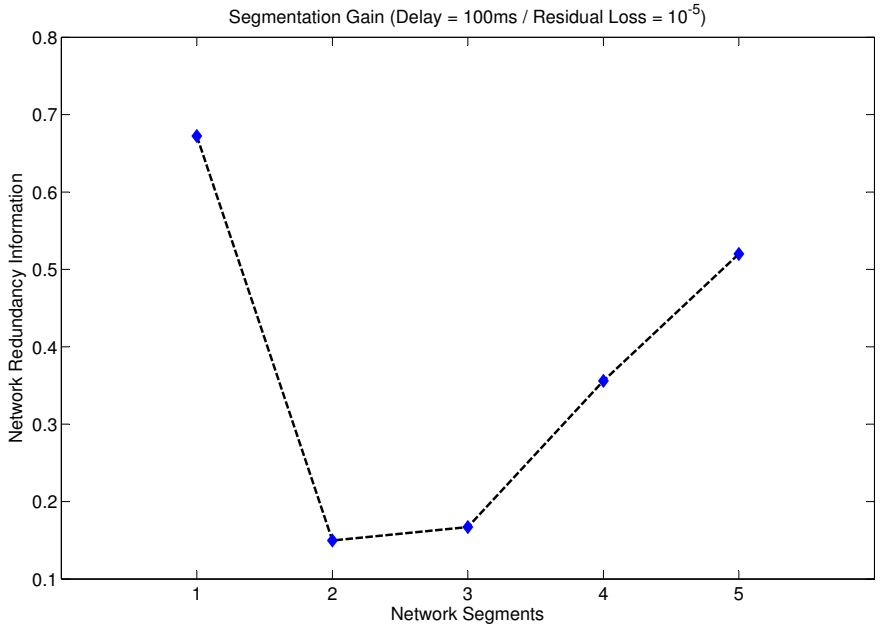

FIGURE 5.10: The AHEC saturation point.

It states, that up to a specific number of segments the amount of required redundancy decreases. This number significantly depends on the parameter of the application and the network infrastructure, i.e. transmission delays and thresholds. In case the number S_0 is exceeded, additional segments do not improve the transmission performance, but increase the required redundancy information. Therefore, an additional segment directly consumes at least the time for its minimum delay configuration. This time subsequently reduces the overall time reserve Γ.

The HARQ *saturation point phenomenon* can be intuitively described: In case the available end-to-end time requirement Δ is significantly larger than the end-to-end path round trip time, the separation of network path segments is a suitable approach. Each segment is individually handled by an error correction approach with sufficient operation time. That is, the HARQ can use optimum configurations and lower the required amount of redundancy. A suitable portion of time can be assigned to the individual segments in order to achieve a globally-optimized error correction performance. In case the number of segments increases, the available time decreases since at least the minimum time $\delta_{i,min}$ is required on each segment i. If the number of segments exceeds the saturation point, the resulting available time is not adequate to cover each segment with the minimum delay configuration, as the transition to the next configuration would consume more time than available. The number and placement of segments on a transmission path is critical. Figure 5.10 illustrates the saturation point for a five segment environment [R1]. Chapter 6 introduces multiple different approaches to find efficient intersection nodes inside a network.

5.4 Viability Considerations

An LDS approach requires a variety of interactions between the data source and the network. The network must contain special relay devices that are able to terminate network traffic, apply error correction schemes and establish a new connection towards the next relay or the designated data sink. The LDS infrastructure must handle the maximum allowed delivery time and residual error rate distribution. The termination and recoding process must be performed transparently to avoid modifications at the original end devices. In chapter 6, the LDS principle is extended by a set of approaches to find optimum locations for path segmentation.

Current networks are generally built using closed-source devices and can hardly be extended to support LDS out of the box. These devices are neither capable of performing such generic error correction operations, nor allow an individual resource assignment as with the budget distribution problem. It is challenging to implement the LDS approach in the unmanaged Internet for the following reasons:

- **Network Knowledge:** Despite having the capabilities to measure individual segment parameters like latency and loss probability, the large variety of network equipment vendors makes it difficult to collect the required information across vendor borders.
- **Application Knowledge:** If the application initiating the connection has special requirements for the transmission, it is challenging to announce those to the network.
- **Specialized Hardware:** Networking hardware is highly specialized to support switching and routing actions. Generic operations as error correction or transparent protocol termination are not desired in general.
- **Interaction:** *Loss Domain Separation* requires atomic computation devices inside the network. These devices must adapt their operations to a global resource distribution in case end-to-end requirements are present. Ongoing traffic must be redirected to these devices and the packet headers must be modified in order to ensure that traffic is accepted by the corresponding network stacks. The network devices must act on behalf of an instance having global knowledge.

The LDS approach breaks with the basic ISO/OSI networking model [38] as the application layer directly influences the layers down to the transport layer. The strict adherence to the OSI layer model dramatically decreases the transmission performance and flexibility. In [R1], a flexible SDN-based loss domain separation mechanism is presented,

that uses a TCP coding relay for network segmentation and demonstrates the advantage of a split transmission compared to a default end-to-end scheme.

A possible application scenario describes the use inside *autonomous systems (AS)* [114]. An AS generally represents a fully managed network maintained by an *Internet Service Provider (ISP)*. In this case, the networking administration has the ability to completely control the network including all required information. The network has predefined ingress and egress nodes that optimize the transmission within this individual AS domain and is therefore completely transparent to the end devices. A second scenario is the application in closed data-centers where operators use new networking technologies. *Google* introduced its backbone network based on the new networking paradigm *Software Defined Networking (SDN)* [149]. SDN technology enables a large set of new features and possibilities for networking, and hence represents an interesting foundation for network engineers. Meanwhile, a set of network device vendors and companies are switching their network infrastructure towards SDN technologies as described in [150, 151].

5.5 Conclusion

This chapter discussed the *Loss Domain Separation (LDS)* principle. Network infrastructures are generally inhomogeneous and consist of a multitude of different network segments. The default application of an end-to-end transmission control including error correction schemes is considered to be suboptimal. With LDS, the network environment is segmented according to the network's parameters and areas of approximately equal characteristics are grouped together. The application of error correction schemes to group segments with similar conditions decreases the amount of required redundancy and reduces the delivery latency. This consequently leads to a multi-hop transmission, consisting of multiple individual segments.

Theoretical aspects show that in case of non-time constraint scenarios a multi-hop transmission always outperforms the traditional end-to-end scheme. Based on Shannon's coding theorem, it has been shown that independent of any specific error correction approach, a maximum number of intersections lowers the required redundancy in the network. In case of time-constraint transmission, retransmission schemes and hybrid error correction mechanisms must be carefully applied, as segmentation has a significant influence on the distribution of the available time budget. As a consequence, a large number of segments severely reduces the coding efficiency.

Chapter 6

Intersection Locations

Transforming an end-to-end transmission path into a segmented path requires knowledge about interior network characteristics. Splitting transmission paths at arbitrary positions may lead to suboptimal results. It is consequently required to map real network properties to comparable values in order to obtain reliable relations between segments. The focus is on interior network nodes as they represent potential intersection locations. A split on a single physical link is considered to be not possible.

Loss domain separation (LDS), as discussed in chapter 5, requires an adequate process of determining the optimum network locations for path segmentation. In the following, concepts based on different characteristics are presented that find suitable multi-hop network path distributions for LDS.

6.1 Intersection Locations: A General Approach

The mapping of a network characteristic to comparable measures is generally achieved by a metric M. The terms metric and characteristics have been discussed in detail in section 4.4. It is abstractly defined as follows:

- Input: Any accessible node characteristic X.
- Output: $M : X \mapsto \mathbb{R}_0^+$.

The *output* is limited to any non-negative real number. The *input* varies and may be any characteristic that can be accessed or measured on a node or segment and can be grouped into two domains:

- *Node specific*: The input of M is directly related to a node. For example, a node specific characteristic is the *in- or out-degree*, the *betweenness*[1] of a node, as presented in section 6.2.2, or a graph characteristic, e.g. *vertex-connectivity* [139].
- *Segment specific*: The input of M is related to the segment's origin or destination. It can be differentiated between *measured* and *statistical* information such as propagation delay or packet loss probability, respectively. Segment specific input requires sharing knowledge between adjacently connected nodes.

It is also possible to combine node and segment specific information into one unified metric input. In section 4.4 it has been shown that merging multi-dimensional metric characteristics into a single metric value is an advantageous mechanism for optimization tasks. In some cases, it can be recommended to separate different inputs to individual metrics and finally evaluate them in a generic approach. Thereby, an iterative procedure is used to incrementally check a set of different metrics. This process is presented in the following.

Assume a transmission path with n interior network nodes N_j and a set of m different metrics M_i. Equation (6.1) defines Λ as a matrix with $dim\,(\Lambda) = (m, n)$, that contains a set of m metrics for n nodes.

$$\Lambda = \begin{pmatrix} \lambda_{1,1} & \lambda_{1,2} & \dots & \lambda_{1,n} \\ \lambda_{2,1} & \lambda_{2,2} & \dots & \lambda_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ \lambda_{m,1} & \lambda_{m,2} & \dots & \lambda_{m,n} \end{pmatrix} \tag{6.1}$$

Hence, the number of rows corresponds to the number of available metrics and the number of columns reflects the number of nodes. $\lambda_{i,j} = M_i(N_j)$ represents the application of metric function M_i at node N_j. In case a metric M_i cannot be evaluated at a node N_j, the entry $\lambda_{i,j} = M_i(N_j)$ is set to zero. Λ intuitively represents the transmission path and the metric-evaluated characteristics as illustrated in figure 6.1, that depicts a scenario with $i = j = 3$. Equation (6.2) shows the intuitive arrangement of metric functions to individual nodes in a matrix Λ.

$$\Lambda = \begin{pmatrix} M_1(N_1) & M_1(N_2) & M_1(N_3) \\ M_2(N_1) & M_2(N_2) & M_2(N_3) \\ M_3(N_1) & M_3(N_2) & M_3(N_3) \end{pmatrix} \tag{6.2}$$

[1] [sic], compare [51]

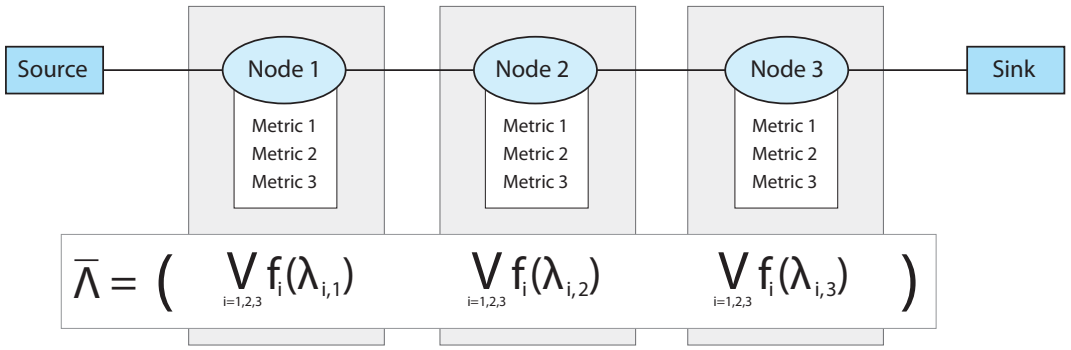

FIGURE 6.1: Finding intersection locations with metrics.

Assume a threshold value $T_i \in \mathbb{R}^+$ for each metric M_i. $M_i > T_i$ implies that the metric's input has been of poor quality and rated as *disadvantageous*. Otherwise the input is regarded as *advantageous*. It is possible to introduce multiple thresholds

$$T_{i,1}, T_{i,2}, \ldots \in \mathbb{R}^+ \tag{6.3}$$

in order to create *sectors* for the evaluation of M_i. Sectors may represent different evaluation areas with gradual quality measures, such as *good*, *moderate*, or *poor*. With definition (6.2), it is also possible to compare metric values of adjacent nodes, such as $M_i(N_j)$ and $M_i(N_{j+1})$.

Let $f_i : \mathbb{R}^+ \mapsto \{0, 1\}$ be the decision function for metric M_i, as generally defined in equation (6.4).

$$f_i(\lambda_{i,j}) = \begin{cases} 1, & \text{intersection at node } N_j \text{ is suggested by metric } M_i \\ 0, & \text{otherwise} \end{cases} \tag{6.4}$$

The specific implementation of this decision function is significantly related to the application scenario. As an example, f_i can be defined to check the metric value $\lambda_{i,j}$ against a threshold value T_i, as shown in equation (6.5). A practical application could be the detection if a link's loss probability is too high.

$$f_i^{\text{example}}(\lambda_{i,j}) = \begin{cases} 1, & \lambda_{i,j} > T_i \\ 0, & \text{otherwise} \end{cases} \tag{6.5}$$

In case $f_i(\lambda_{i,j})$ is applied to the elements of Λ, a new matrix Λ_{eval} with entries $\lambda_{i,j}^{eval} = f_i(\lambda_{i,j})$ is obtained. Each column represents the evaluation of all available

metrics at a certain node N_j. In analogy to equation (6.2), equation (6.6) presents a sample of Λ_{eval}.

$$\Lambda_{eval} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \end{pmatrix} \tag{6.6}$$

It can be seen that metric M_3 at node N_1, as well as metrics M_2 and M_3 at node N_2, are evaluated to $1s$. Consequently, these metric values exceeded the predefined threshold and imply the application of a segmentation at these nodes. A path separation at a node is suitable if it contains at least one 1 in the corresponding column. This means at least one metric has a value that implies a split at this location. For practical approaches, the number of $1s$ in a column, that indicate a segmentation, may vary. In case a single 1 in a column is sufficient, the individual application of the *logical-or* operator $\bigvee$ to each column is possible, as shown in equation (6.7) and figure 6.1.

$$\begin{aligned} \bar{\Lambda} &= \begin{pmatrix} \bigvee_i f_i(\lambda_{i,1}) & \bigvee_i f_i(\lambda_{i,2}) & \dots & \bigvee_i f_i(\lambda_{i,n}) \end{pmatrix} \\ &= \begin{pmatrix} \lambda_1 & \lambda_2 & \dots & \lambda_n \end{pmatrix} \end{aligned} \tag{6.7}$$

$\bar{\Lambda}$ is a row vector of dimension $(1 \times n)$ with binary entries 0 or 1. Note that each entry $\bar{\lambda}_j$ represents exactly one node on the network path and according to this intuitive scheme it holds

$$\bar{\lambda}_j = \begin{cases} 0, & \text{no split required at node } N_j \\ 1, & \text{split required at node } N_j \end{cases} \tag{6.8}$$

This approach provides a scalable, fast, and reliable method to check if an interior node is an appropriate location to split a transmission path. It is highly flexible to allow individual decision criteria with a deterministic outcome.

6.2 General Location Concepts

The location assignment is an important process that determines where LDS relays are placed inside a network. It defines the operating point of the network optimization and consequently influences the efficiency. The main objective focus is on a location assignment for efficient operations depending on the operation mode. In either case multiple principal assignment approaches are possible. In the following, *strategic* and *network topology based*, as well as LP-based *optimized* location assignment schemes are discussed.

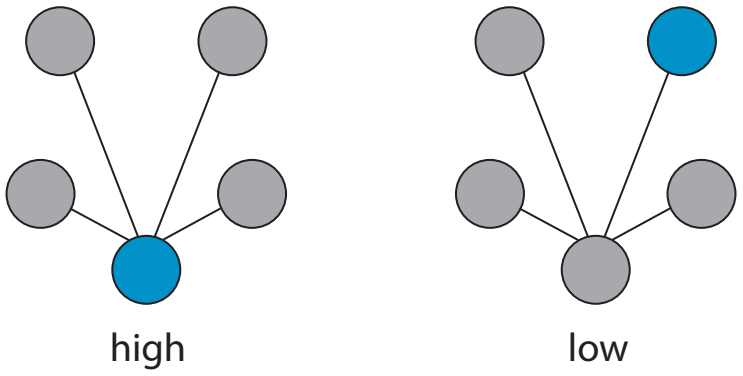

FIGURE 6.2: The node degree metric.

6.2.1 Strategic Assignment

This strategic assignment concept refers to a location selection based on non-technical criteria and mainly focuses on economic aspects and policies. The LDS split locations are determined without any reference to the present network topology or state. These subjective assignments rely on factors such as hardware capacity, financial objectives, customer requirements, or networking policies. Strategic assignments are restricted to individual administrative domains, such as *Autonomous Systems (AS)* [114]. As an example, LDS intersections are located at ingress and egress nodes of an AS, as it enables the adjustment of transport data for a suitable transmission inside the AS.

6.2.2 Topology Based Assignment

A metric-based LDS segment location assignment uses knowledge about the network topology. It does not contain information about the current state. *Network topology based* assignments utilize graph theoretical algorithms, characteristics, and mechanisms [14]. Graph theory additionally provides approaches for network analysis focusing on the topology at a higher level. A typical example is the *k-vertex-connected graph* [139]. This characteristic describes how many nodes of a graph must be removed in order to split it into multiple graphs. Another prominent description of graph theory are *cliques* [139]. They represent parts of the topology where all nodes are individually connected. All of these descriptions can be used to find an appropriate location of LDS relays in a network. The application scenario and the functionality influence the split locations. In the following, prominent representations for topology characteristics are presented.

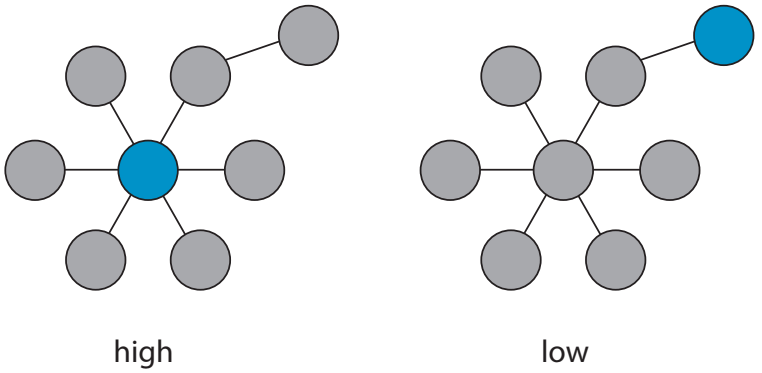

FIGURE 6.3: The node closeness metric.

Degree Centrality

The node *degree centrality* [51] represents the number of connections of a node to other nodes in a network as illustrated in figure 6.2. The higher the degree of a node, the more nodes are directly connected to it. The node degree can be used as an indicator for the probability of occurrence in a distribution scheme such as a single route or a multicast tree. In case the node degree is high, it is likely to have the node included in a dissemination scheme. Assuming an adjacency matrix X with entries $x_{ij} = \{0, 1\}$, where $x_{ij} = 1$ in case node n_i is connected to node n_j. Otherwise, it holds $x_{ij} = 0$. The *degree* of a node n_i is then defined as shown in equation (6.9).

$$\text{degree}(n_i) = \sum_j x_{ij} \tag{6.9}$$

Closeness Centrality

The *node closeness centrality* [51] represents the node's distance relation within the network. It is based on the sum of the shortest path distances to all other nodes and reflects how close a node is to all other nodes in the network. Larger distances are assumed to introduce more costs and consume more network resources. Figure 6.3 illustrates the node closeness metric. In case d_{ij} states the distance between node n_i and n_j, the *closeness* measure of a node n_i is defined as shown in equation (6.10).

$$\text{closeness}(n_i) = \frac{1}{\sum_j d_{ij}} \tag{6.10}$$

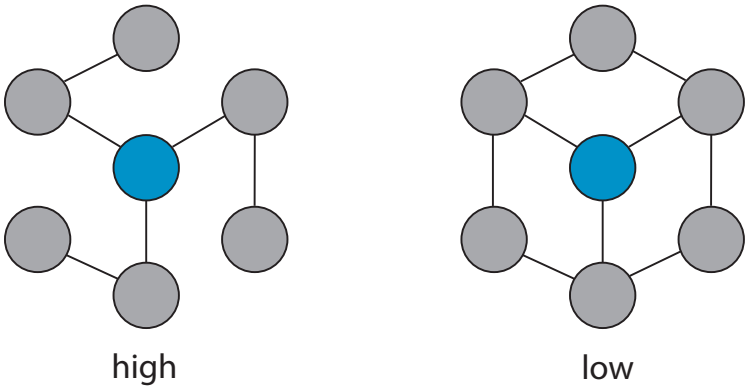

FIGURE 6.4: The node betweenness metric.

Betweenness Centrality

The *node betweenness*[2] *centrality* [51] of a node n_i refers to the number of shortest paths between any two nodes in the network that directly cross n_i. A large value reflects a topology where the node is centrally located within the network. Consequently, it represents a measure of how much a specific node controls the flows between any other two nodes. Figure 6.4 illustrates the node betweenness metric. Assuming that $|SP|$ represents the total number of shortest paths in the network, and $|SP|_i$ denotes the number of shortest paths crossing node n_i. The *betweenness* measure for node n_i is defined as shown in equation (6.11).

$$\text{betweenness}(n_i) = \frac{|SP|_i}{|SP|} \tag{6.11}$$

6.2.3 Optimized Assignment

An optimized assignment principle does not solely rely on subjective and static objective characteristics. It additionally considers the global environmental structure and uses a mathematical model that precisely unveils adequate locations. In the following a *Linear Programming* optimization scheme is presented, that solves the location problem for *error correction relays*. The model refers to the general *Linear Programming (LP)* model as introduced in section 4.2.1.

In the following the unicast routing LP model from section 4.2.1 is extended to find error correction relay locations according to the LDS principle. For this LP formulation a single unicast stream and the presence of one relay at each node $n_i \in V$ is assumed. Note, that these relays may either be active or not.

[2][sic], compare [51]

The LP model for the error correction relay location problem is formulated as follows [137]:

$$\min \sum_{(i,j)\in E} (w_{ij} \cdot x_{ij} + \gamma_i \cdot y_i) \tag{6.12}$$

Subject to:

$$\sum_{(i,j)\in E_i^-} x_{ij} - \sum_{(j,i)\in E_i^+} x_{ji} = b_i \qquad \forall i \in V \tag{6.13}$$

$$b_i = \begin{cases} 1, & \text{if } i \text{ is the source,} \\ -1, & \text{if } i \text{ is the sink,} \\ 0, & \text{otherwise.} \end{cases} \qquad \forall i \in V \tag{6.14}$$

$$x_{ij} \cdot d - c_{ij} \leq 0 \qquad \forall (i,j) \in E \tag{6.15}$$

$$\sum_{(i,j)\in E} x_{ij} \cdot \lambda_{ij} + \sum_{(i,j)\in E} r_{ij} \cdot \theta_i \leq \Lambda \tag{6.16}$$

$$r_{ij} - x_{ij} \leq 0 \qquad \forall (i,j) \in E_k^- \tag{6.17}$$

$$r_{ij} - y_i \leq 0 \qquad \forall (i,j) \in E_k^- \tag{6.18}$$

$$x_{ij} + y_i - r_{ij} \leq 1 \qquad \forall (i,j) \in E_k^- \tag{6.19}$$

This model extends the unicast routing model presented in section 4.2.1. The corresponding flow balance in equations (6.13) and (6.14) as well as the capacity constraint in equation (6.15) remain unchanged.

The objective function (6.12) is extended by a second term that represents the costs introduced by an activated LDS relay. As not all relays are active, the indicator variable y_i represents the current relay status at node n_i as shown in equation (6.20).

$$y_i = \begin{cases} 1, & \text{relay is active} \\ 0, & \text{otherwise} \end{cases} \tag{6.20}$$

$y_i = 0$ implies a state where either no relay is available or the relay is present but inactive at node n_i. For the proposed model both cases are assumed to be equivalent.

Equation (6.16) represents a general constraint that must be met by the selected route. The sum over all $x_{ij} \cdot \lambda_{ij}$ represents the accumulation of the link's characteristic λ chosen to be constraint, such as the link's latency or packet jitter. The multiplication with x_{ij} avoids the summation of link characteristics that are not on the selected route. The second term in equation (6.16) states the improvement with this constraint when using a

relay. θ_i reflects this improvement and is assumed to be negative. Following the example from above, θ_i can be a latency reduction due to the utilization of a relay. This may be implemented using sophisticated buffer structures or shorter packet retransmission cycles as proposed in chapter 5. θ_i is multiplied with the variable r_{ij}. It indicates if the relay at node n_i is active and the egress link (i, j) is used with at least one unit of flow. In Boolean logic it can be expressed as $r_{ij} = x_{ij} \wedge y_i$. Linear Programming allows the use of Boolean algebra as shown in [135]. Equations (6.17) - (6.19) represent the corresponding logic for r_{ij}. Consequently, only if the relay is active and flow is available, the improvement from the relay can be included and considered in the constraint sum of equation (6.16). As it is a constraint, the sums on the left hand side of the equation are strictly limited by Λ. It represents an upper bound for this characteristic, such as the maximum allowed transmission latency for the whole unicast path.

As linear programs are powerful tools to optimize network transmission, the next section presents fundamental techniques to model network environments with both, *additive* and *multiplicative* characteristics.

6.3 Network Characteristics and Linear Programs

A *characteristic* represents a numerical measure for quality. In the domain of networking this can be a measure to rate the quality of a path or a node. This quality value is used as a decision base for routing algorithms. A link's *latency* or *loss probability* is a typical network characteristic that can be used for routing decisions.

Characteristics can be categorized into two groups. *Additive* characteristics represent a set of individual values that can be accumulated by building their mathematical sum. *Multiplicative* characteristics are accumulated by building their product. Both types can generally not be merged. *Latency* corresponds to *additive* and *packet loss probability* to *multiplicative* characteristics. The accumulation of a set of characteristics is required when considering network paths. Paths are built by consecutive links, each with its own characteristic value. The main representative characteristic for the whole path is therefore the accumulation of all individual values.

In an individual application this does not lead to difficulties in case the corresponding characteristic's nature is considered during design and implementation. A *Linear Program (LP)* model uses indicator variables and assumes a linear combination of them. This avoids the use of multiplicative characteristic values in LPs. A common way of transforming a product of values into a sum of values is provided by the *logarithmic identities* [144].

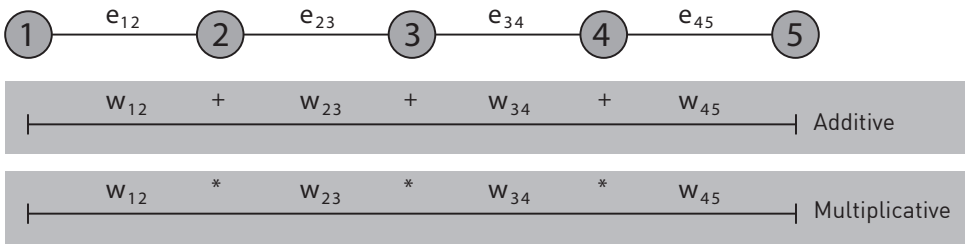

FIGURE 6.5: Additive and multiplicative path characteristics.

In the following both characteristic types are discussed and the application of *logarithmic identities* is presented. This enables the representation of multiplicative characteristics in *Linear Programs*. The application of network characteristics as a metric value is assumed.

6.3.1 Additive Characteristics

Additive implies that the individual characteristics can be linearly accumulated as a sum. First, assume a set of links e_{ij} and corresponding individual weights $w_{ij} = \text{weight}(e_{ij})$. Figure 6.5 shows a path with four links $e_{12}, e_{23}, e_{34}, e_{45}$ and additive weights. The weight between node 1 and 5 is calculated as the sum of all individual weights w_{ij} as shown in equation (6.21).

$$w_{15} = w_{12} + w_{23} + w_{34} + w_{45} \tag{6.21}$$

The scenario illustrated in figure 6.6 represents a network with two paths between nodes 1 and 4.

In an LP model the indicator variables x_{ij} are used to differentiate between active and non-active links. In case the path from 1 to 4 via node 2 has been selected, it eventually holds $x_{12} = x_{24} = 1$ and $x_{13} = x_{34} = 0$. The assumed additive characteristic allows a straightforward application considering the link weights. As active links have an indicator variable set to 1, the weight for the path between node 1 and 4 is calculated as shown in equation (6.22).

$$\begin{aligned} w_{14} &= \sum_i \sum_j x_{ij} \cdot w_{ij} \\ &= x_{12} \cdot w_{12} + x_{24} \cdot w_{24} + \underbrace{x_{13} \cdot w_{13}}_{=0} + \underbrace{x_{34} \cdot w_{34}}_{=0} \\ &= w_{12} + w_{24} \end{aligned} \tag{6.22}$$

The non-active indicator variables $x_{13} = x_{34} = 0$ eliminate the non-essential metric values. Under the assumption that the LP model correctly selects active edges, it is

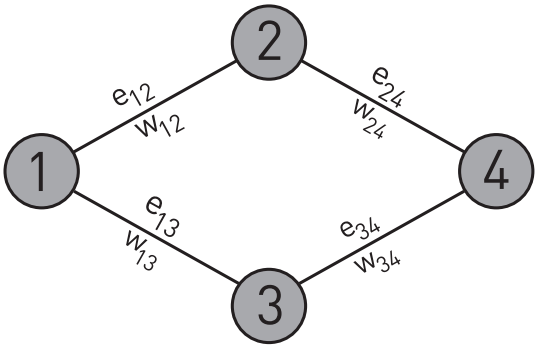

FIGURE 6.6: Example network for characteristics demonstration.

possible to sum up the products of indicator variables and the corresponding edge metric values. *Additive* metrics are applicable for LP models with no modification and provide a suitable basis of transport network models. They have already been used throughout this thesis, e.g. in section 4.2.1.

6.3.2 Multiplicative Characteristics

Multiplicative characteristics must be transformed when used in *Linear Programs*. Assume a set of links e_{ij} and corresponding individual weights w_{ij}. Figure 6.5 also shows a path with four links e_{12}, e_{23}, e_{34}, and e_{45} with multiplicative weights. Such a path with individual *multiplicative characteristics* can also be seen as a system chain with individual components. As an example, the illustrated chain may represent a telecommunication system and the individual characteristics are *noise figures* [152]. The accumulated total weight of the processing chain, as depicted in figure 6.5, is calculated as shown in equation (6.23).

$$w_{15} = w_{12} \cdot w_{23} \cdot w_{34} \cdot w_{45} \tag{6.23}$$

In a more general case with a set of n weights w_i, the overall accumulated weight is obtained by equation (6.24).

$$w_{total} = \prod_{i=1}^{n} w_i \tag{6.24}$$

Due to the multiplication, an individual value w_i directly influences the total value significantly. In case the values are defined within a well-known range, it is not possible to correct an individual low value with a high one as it is possible with additive metrics. *Linear Programs* solely require additive weights in both, objective functions and constraint equations. A set of *multiplicative* characteristics must be transformed to a set of values allowing a linear combination of their individual components. The *logarithm*

function in combination with a *logarithmic identity* [144] transforms a product into a sum as shown in equation (6.25).

$$\begin{aligned} w_1 \cdot w_2 \cdot \ldots \cdot w_n &= \prod_{i=1}^{n} w_i \\ \Leftrightarrow \log\left(w_1 \cdot w_2 \cdot \ldots \cdot w_n\right) &= \log\left(\prod_{i=1}^{n} w_i\right) \\ &= \sum_{i=1}^{n} \log\left(w_i\right) \end{aligned} \tag{6.25}$$

Figure 6.6 shows a network with multiple potential paths. The set of active edges is obtained at the end of the optimization process modeled by an LP.

It is required to use a representation that enables the application in an LP model. The challenge is to model these constraints in a way that non-active links do not contribute to the weight value that is used for decisions. In case of *additive* characteristics, equation (6.22) shows, that links with non-active indicator variables set their contribution to zero.

For networks with multiplicative characteristics it is not possible to multiply indicator variable and weight. Consequently, a new representation is required. Consider figure 6.6 and assume that the path via nodes $1, 2, 4$ is selected. As indicator variables x_{ij} are either 0 or 1, they are used as exponents to the weight values to achieve reliable total end-to-end weight measure w_{14} as shown in equation (6.26). The selected path implies $x_{12} = x_{24} = 1$ and $x_{13} = x_{34} = 0$ leading to $w_{12}^{x_{12}} = w_{12}$ and $w_{24}^{x_{24}} = w_{24}$ while $w_{13}^{x_{13}} = w_{34}^{x_{34}} = 1$.

$$\begin{aligned} w_{14} &= w_{12}^{x_{12}} \cdot w_{13}^{x_{13}} \cdot w_{24}^{x_{24}} \cdot w_{34}^{x_{34}} \\ &= w_{12}^{1} \cdot w_{13}^{0} \cdot w_{24}^{1} \cdot w_{34}^{0} \\ &= w_{12} \cdot w_{24} \end{aligned} \tag{6.26}$$

Equation (6.26) represents a valid representation for this kind of characteristic values.

The assignment in an LP model requires the application of the logarithm to transform the product into a sum as shown in equation (6.27).

$$\begin{aligned} w_{14} &= w_{12}^{x_{12}} \cdot w_{13}^{x_{13}} \cdot w_{24}^{x_{24}} \cdot w_{34}^{x_{34}} \\ \Leftrightarrow \log\left(w_{14}\right) &= \log\left(w_{12}^{x_{12}} \cdot w_{13}^{x_{13}} \cdot w_{24}^{x_{24}} \cdot w_{34}^{x_{34}}\right) \\ &= \log\left(w_{12}^{x_{12}}\right) + \log\left(w_{13}^{x_{13}}\right) + \log\left(w_{24}^{x_{24}}\right) + \log\left(w_{34}^{x_{34}}\right) \\ &= x_{12} \cdot \log\left(w_{12}\right) + x_{13} \cdot \log\left(w_{13}\right) + x_{24} \cdot \log\left(w_{24}\right) + x_{34} \cdot \log\left(w_{34}\right) \end{aligned} \tag{6.27}$$

In case the end-to-end value of w_{14} is limited by a threshold $\Lambda \geq w_{14}$ the LP restriction can be modeled as

$$\begin{aligned} \log(\Lambda) &\geq \log(w_{14}) \\ &= x_{12} \cdot \log(w_{12}) + x_{13} \cdot \log(w_{13}) + x_{24} \cdot \log(w_{24}) + x_{34} \cdot \log(w_{34}) . \end{aligned} \tag{6.28}$$

Note, that the direction of the inequality $\Lambda \geq w_{14}$ remains unchanged as the equivalent transformation operates with the logarithmic function, that is strictly monotonically increasing. The last line of equation (6.27) shows that the indicator variables x_{ij} are multiplied with a simple numeric constant $\log(w_{ij})$. Consequently, this transformation and formulation approach enables a valid assignment of multiplicative metrics in LP models.

It is important to mention that the logarithm function turns values between $0 < w_{ij} < 1$ into the negative domain and values between $1 < w_{ij} < \infty$ into the positive domain. In case $\exists w_i \in]0, 1[$ and $\exists w_j \in]1, \infty[$, the summation is corrupted due to the mixture of positive and negative values after the application of the logarithm. To avoid such incidents, it is requested to normalize all weights prior the logarithmic transformation to a single value space, that is $]0, 1[$ or $]1, \infty[$.

Hence, in case the weights represent probabilities, it is assumed that $\Lambda \in]0, 1[$ and $w \in]0, 1[$.

6.3.3 LP Extension for the Error Correction Relay Model

In section 6.2.3 the *Error Correction Relay* model based on *Linear Programming* optimization has been presented. The presented model incorporates additive characteristics, such as latency. In this section the model from section 6.2.3 is extended to utilize multiplicative characteristics, such as packet loss probability.

As shown in section 6.3.2, individual multiplicative characteristics directly influence the complete path weight. In analogy to equation (6.16) the corresponding constraint with multiplicative character can be represented as shown in equation (6.29).

$$\sum_{(i,j) \in E} x_{ij} \cdot \log(w_i) + \sum_{(i,j) \in E} r_{ij} \cdot \log(\Theta_i) \leq \log(\Lambda) \tag{6.29}$$

Note that $0 < \Theta_i \leq 1$ is a result of the logarithm application. In case $\Theta_i = 1$ no benefit is achieved. $\Theta_i = 0$ would contradict a logical assumption where error correction relays cannot be used without any costs. Consequently, the sum of all $r_{ij} \cdot \log(\Theta_i)$ is subtracted from the accumulated characteristics as $\log(\Theta_i) \leq 0$ for $0 < \Theta_i \leq 1$.

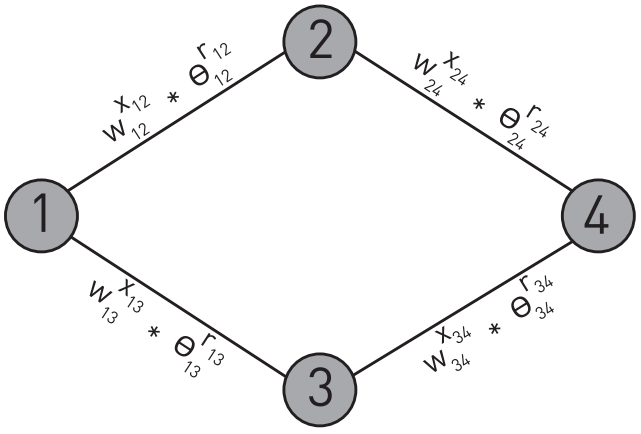

FIGURE 6.7: Network with multiplicative characteristics for the relay location problem.

It holds:

$$
\begin{aligned}
&\prod_{(i,j)\in E} \left(w_i^{x_{ij}} \cdot \theta_i^{r_{ij}}\right) && \leq \Lambda \\
\Leftrightarrow &\log\left(\prod_{(i,j)\in E} \left(w_i^{x_{ij}} \cdot \theta_i^{r_{ij}}\right)\right) && \leq \log\left(\Lambda\right) \\
\Leftrightarrow &\sum_{(i,j)\in E} \log(w_i^{x_{ij}} \cdot \theta_i^{r_{ij}}) && \leq \log(\Lambda) \\
\Leftrightarrow &\sum_{(i,j)\in E} \log(w_i^{x_{ij}}) + \sum_{(i,j)\in E} \log(\theta_i^{r_{ij}}) && \leq \log(\Lambda) \\
\Leftrightarrow &\sum_{(i,j)\in E} x_{ij} \cdot \log(w_i) + \underbrace{\sum_{(i,j)\in E} r_{ij} \cdot \log(\theta_i)}_{\leq 0} && \leq \log(\Lambda)
\end{aligned}
\tag{6.30}
$$

The logarithm is a strictly monotonic increasing function, the unequal signs must not be inverted when it is applied. As $0 < \Theta \leq 1$ it follows that $-\infty < \log\left(\Theta\right) \leq 0$.

Consider figure 6.7, which illustrates a sample network with multiplicative characteristics w and relay benefit values Θ. Assume a selected path $(1,2),(2,4)$ with an activated relay at node 2. Following equation (6.20) it holds $y_2 = 1$. It follows $x_{12} = x_{24} = 1$ and $x_{13} = x_{34} = 0$. With $r_{ij} = x_{ij} \wedge y_i$ it consequently holds $r_{12} = r_{13} = r_{34} = 0$ and $r_{24} = 1$.

For the total weight it is seen that only the active edges are considered, as shown in equation (6.31).

$$
\begin{aligned}
w_{total} &= w_{12}^{x_{12}} \cdot \theta_{12}^{r_{12}} \cdot w_{24}^{x_{24}} \cdot \theta_{24}^{r_{24}} \cdot w_{13}^{x_{13}} \cdot \theta_{13}^{r_{13}} \cdot w_{34}^{x_{34}} \cdot \theta_{34}^{r_{34}} \\
&= w_{12}^{1} \cdot \underbrace{\theta_{12}^{0}}_{=1} \cdot w_{24}^{1} \cdot \theta_{24}^{1} \cdot \underbrace{w_{13}^{0} \cdot \theta_{13}^{0} \cdot w_{34}^{0} \cdot \theta_{34}^{0}}_{=1} \\
&= w_{12} \cdot w_{24} \cdot \theta_{24}
\end{aligned}
\tag{6.31}
$$

This representation eventually includes the selected links and their weights as well as presents relay benefits in the total weight.

Equation (6.29) can be used analogously to equation (6.16) in an LP model, as presented in section 6.2.3, to express a constraint handling of multiplicative weight values.

6.4 Efficient Resource Distribution

Multimedia data has strict transmission requirements as it is highly sensitive to *latency*, *packet loss*, or *jitter*. If these are encountered between the sender and receiver, they are called *global* requirements and the network between the end nodes is abstracted to be a single virtual link. Such characteristics are measurable and have a corresponding acceptable upper limit that must not be exceeded in order to achieve an acceptable QoS and QoE level [40, 9].

Traditional transmission schemes are implemented with end-to-end characteristics. In this case, the global requirements must be met between the end nodes of the transmission. The connection itself is assumed to be *virtually direct*, which means all segments en route are merged into a single *logical* link. As only one segment in this scenario is traceable, the global requirements *budget* can be assigned as a compact unit to this single segment. This approach requires knowledge and transmission interactions only at the end nodes of the path. In case of a split transmission path, multiple segments have to be considered instead of a single link. This implies a new challenge as the global requirements have to be distributed over the segments of the path. In case of a latency requirement a corresponding reasonable upper latency limit exists that simultaneously represents the global latency budget. With multiple segments, this budget must also be split and assigned to individual segments. In case there are two segments and the time budget is 500 ms, each segment can have assigned a portion of these 500 ms, but the sum of both must be at most 500 ms. Latency represents a crucial transmission characteristic as shown in chapter 5. It influences the error correction scheme on the segments according to the assigned portion of the budget. Note, that there generally exists a lower limit for the portion assigned to a single segment. For latency, it can be seen that packets require at least half of the round-trip time (RTT) on each individual segment. This is a fixed parameter and must also be considered when distributing resources on segments.

Assume a global requirement Δ that restricts the end-to-end delivery time. If this budget is split among the segments, each segment gets a smaller portion of the global budget as illustrated in figure 6.8. According to Shannon's *Channel Coding theorem* [35], a smaller

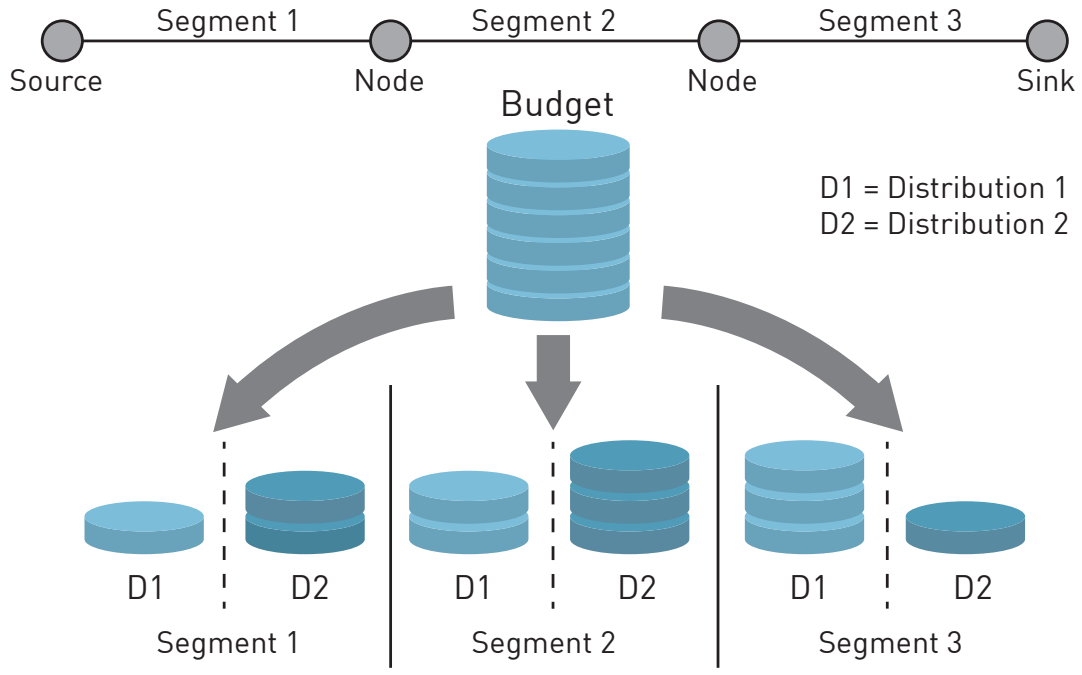

FIGURE 6.8: Exemplary budget distributions for multiple segments.

portion of time results in more redundancy information and higher data rate in case an error correction scheme is applied. A portion of the global budget on a segment is called *atomic budget*.

A feasible time distribution follows the principle that segments with a high probability of error require a larger time budget than segments with smaller error probabilities. The task is to find a feasible trade-off between the size and ordering of atomic budgets to achieve an acceptable error correction performance.

This distribution problem is closely related to the well-known *Knapsack* combinatorial optimization problem [134]. This problem includes a set of items which must be selected according to different characteristics. The solution is to obtain an optimal set of selected items under the condition that a global budget is not exceeded. Multiple combinations may be valid, but the issue is to find a valid and acceptable distribution.

A straightforward approach is to solve this problem by a *greedy* or *naive full* search algorithm [16]. Due to the potentially large number of segments and possible valid distributions on a network path, the problem complexity grows exponentially. As an example for the path length consider the following `traceroute`[3] result from the Saarland University campus network to Akamai Technologies[4]:

[3] http://support.microsoft.com/kb/162326/en/ (2015/01/29)
[4] http://www.akamai.com/ (2015/01/29)

```
 1    <1 ms    <1 ms     1 ms  134.96.40.1
 2    <1 ms    <1 ms    <1 ms  c65eb36-win.net.uni-saarland.de [134.96.6.54]
 3     1 ms    <1 ms    <1 ms  xr-saa1-pc1.x-win.dfn.de [188.1.234.37]
 4     2 ms     2 ms     2 ms  xr-kai1-te2-1.x-win.dfn.de [188.1.145.98]
 5     6 ms     6 ms     6 ms  xr-fzk1-te2-4.x-win.dfn.de [188.1.145.101]
 6     7 ms     7 ms     7 ms  cr-fra1-be9.x-win.dfn.de [188.1.144.121]
 7     6 ms     6 ms     6 ms  xe-1-2-0.mpr1.fra4.de.above.net [80.81.194.26]
 8     6 ms     6 ms     6 ms  ae8.mpr1.fra3.de.zip.zayo.com [64.125.26.233]
 9    12 ms    12 ms    12 ms  ae4.cr1.ams5.nl.zip.zayo.com [64.125.32.106]
10    89 ms    89 ms    90 ms  xe-7-1-0.cr2.lga5.us.zip.zayo.com [64.125.20.89]
11    95 ms    96 ms    95 ms  ae1.cr1.lga5.us.zip.zayo.com [64.125.29.37]
12    89 ms    89 ms    89 ms  ae8.er1.lga5.us.zip.zayo.com [64.125.26.162]
13    89 ms    89 ms    89 ms  23.192.10.10
```

This connection uses 12 different segments for an intercontinental best-effort transmission and requires about 89 *ms* to reach the desired destination.

A more sophisticated solution approach refers to the *Multiple-Choice Knapsack Problem (MCKP)* [153]. The MCKP is a more general version of the *Knapsack Problem (KP)* and can be modified to solve the resource distribution problem as shown in [R8]. The *KP* aims to find a selection of items that generates an optimum outcome while a given capacity constraint is not exceeded. MCKP additionally groups the set of available items into multiple distinct partitions and extends the original constraint by requiring that eventually exactly one item per partition must be selected. The MCKP is schematically shown in figure 6.9.

The MCKP is basically formulated as shown in equation (6.32) [153].

$$\begin{aligned} \text{maximize} \quad & \textstyle\sum_j p_j \cdot x_j & \\ \text{s.t.} \quad & \textstyle\sum_j w_j \cdot x_j & \leq C \\ & \forall k : \textstyle\sum_{i \in N_k} x_i & = 1 \end{aligned} \tag{6.32}$$

In equation (6.32) w_j reflects the weight and p_j the profit of an item i_j in a partition N_k. The total capacity constraint of the *knapsack* is represented by C and x_j is the indicator variable of item i_j:

$$x_j = \begin{cases} 1, & \text{item } i_j \text{ was selected} \\ 0, & \text{otherwise.} \end{cases} \tag{6.33}$$

According to equation (6.33), the indicator variable x_j is used analogously to the LP indicator variables in section 4.2.1.

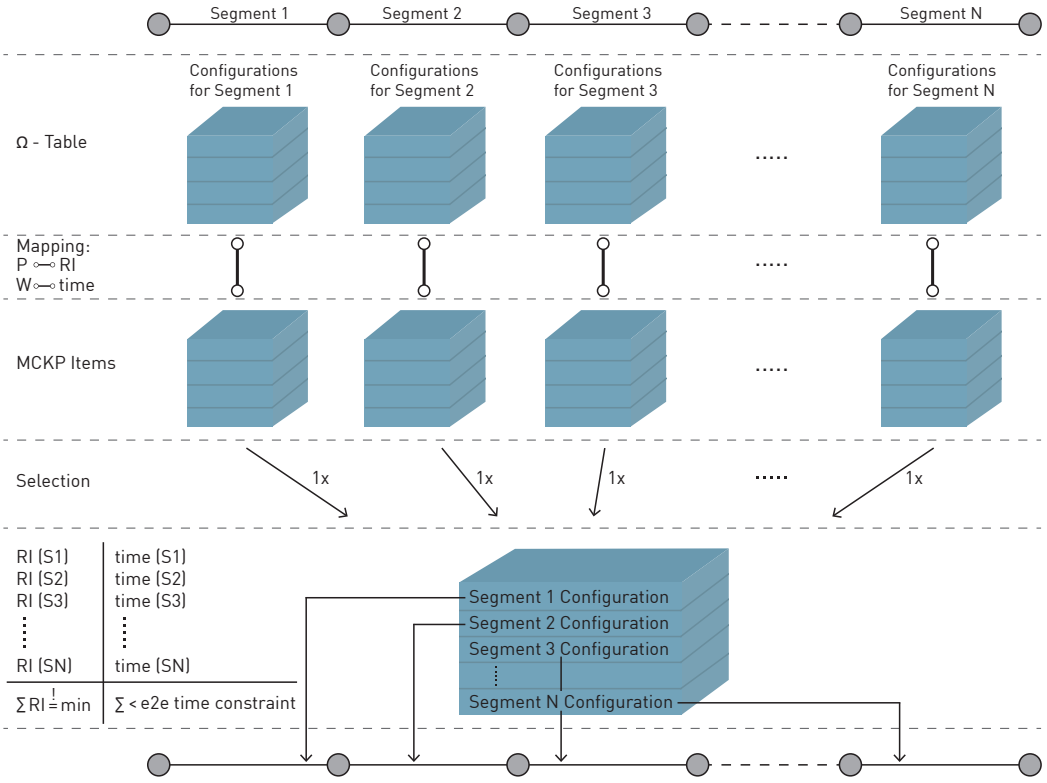

FIGURE 6.9: Multiple-Choice Knapsack Problem (MCKP) for budget distribution.

The basic *Knapsack Problem* can be reduced to the MCKP. Consequently, the MCKP itself is *NP-hard* [153]. The MCKP can be intuitively modified to meet the characteristics of the presented resource distribution problem. Items are considered to be atomic. This means they are predefined units carrying information about the assigned partitions of each global budget. This information is stored in the weight and the profit of an item. For the mapping to the distribution problem, weight corresponds to the amount of used budget and profit to the error correction performance. In case there are multiple budgets, the MCKP would also have multiple weights. Items and their information characteristics may be precomputed and stored in a table or generated online, depending on the environmental conditions. The capacity constraint is replaced by the global budget as an upper limit. The item partitions naturally correspond to the network segments. Table 6.1 presents an overview of the correspondences between MCKP and the budget distribution. The given problem can be modeled as an MCKP and solved with well-known solution algorithms [153]. An exemplary MCKP dynamic programming based solution [153] is presented in [R8], where it has been shown that the search space for this MCKP scenario can be significantly reduced by the determination of dominating error correction configurations.

MCKP	Budget Distribution
Item	Atomic unit that can be assigned Example: Error correction configuration
Item / Profit	Parameter that measures improvement when using this configuration Example: Loss reduction
Item / Weight	Parameter that measures effort when using this configuration Example: Data redundancy
Partition	Set of atomic units for one specific segment Example: Set of error correction parameter
Capacity	Upper end-to-end limit for accumulated atomic unit characteristic Example: Latency

TABLE 6.1: Mapping correspondences between the Multiple-Choice Knapsack Problem (MCKP) and the budget distribution.

6.5 Conclusion

The multi-segment structure of a transmission path is a critical factor with respect to the efficiency. It is possible to identify suitable locations for path segmentation based on different characteristics. In this chapter, a general approach for multi-metric decisions is presented along with a set of basic concepts based on network topology. Additionally, an LP model for optimum locations of error correction relay has been presented, that uses additive characteristics. As network environments may have additive and multiplicative characteristics, a suitable transformation approach has been given to use LP models for both types of characteristics. As the application of multi-hop transmissions with a given time restriction leads to a time budget distribution problem for the individual segments, a Multiple-Choice Knapsack Problem (MCKP) formulation is presented to find efficient multi-segment time distributions.

Chapter 7

Proof of Concept: Demonstrators

This chapter describes two fundamental proof of concept approaches to validate network optimization strategies. It includes a local Wide Area Network (LWAN) demonstrator, that serves as a basis for pure networking demonstrations. It contains a web-based user interface for essential interactions. As this demonstrator is hardly extendable and thus impractical for joint research, it requires an extended set of functionality to become a convenient network testbed. Consequently, the LWAN demonstrator has been further developed and its essential characteristics have been transformed to fit into a Software Defined Network (SDN) environment. Such an SDN testbed offers a large set of new opportunities to network engineers, such as highly flexible traffic interactions. During this thesis, a powerful SDN-based environment has been created at the Saarland University that allows flexible wide area networking transmission experiments. It offers the possibility to share knowledge with other researchers and provides an excellent basis for future research collaborations.

7.1 Local WAN Demonstrator

The development of a *local WAN (LWAN) demonstrator* has been triggered by the need for an individual network testbed to benchmark and enhance the *Predictable Reliability under Predictable Delay (PRPD)* principle [66]. It serves as a proof of concept demonstrator that incorporates functionality on a bare network basis for multi-hop transmissions using UDP [2] and PRRT [66]. The demonstrator is able to simulate environments with restricted and application specific delay and residual loss rate requirements. It allows to interactively modify transmission characteristics as delay and loss on a per

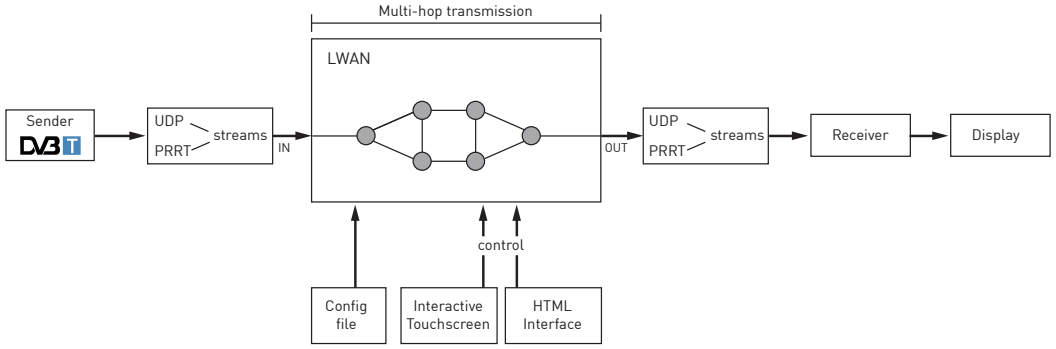

FIGURE 7.1: Schematic overview of the LWAN demonstrator.

link basis and continuously provides a visual evaluation of the network statistics by the help of intuitive diagrams. The LWAN demonstrator represents a testbed for multi-hop transmissions in a small network context built with physical devices. It has been designed in form of a blackbox network system with variable input and output possibilities as illustrated in figure 7.1. Thereby, data is streamed over a set of different routes from a single sender to a receiver. The LWAN actually implements DVB-T[1] and file playback video sources, including stereoscopic content. It is possible to directly display the video stream at the receiver and evaluate its quality. The LWAN demonstrator also allows to generate cross traffic in the network in order to provide realistic test environments.

All of these features can be controlled via an HTML-based interface. It provides functionality to start or stop streams, change transmission routes, modify link characteristics, and evaluate network statistics on the fly. This HTML interface is accompanied by an exhibition wall with integrated touch-sensitive displays. Each display is associated with a certain LWAN node and represents its egress links. It is possible to directly modify the respective link's characteristics by interacting with a display.

This demonstrator has been developed at the Telecommunications Lab at the Saarland University. It is composed of three components: a mobile rack including multiple hardware server machines, a sender/receiver pair, and a customized exhibition wall of dimension 2x3 meters. Figure 7.2 presents the components.

[1] http://www.dvb.org (2015/01/29)

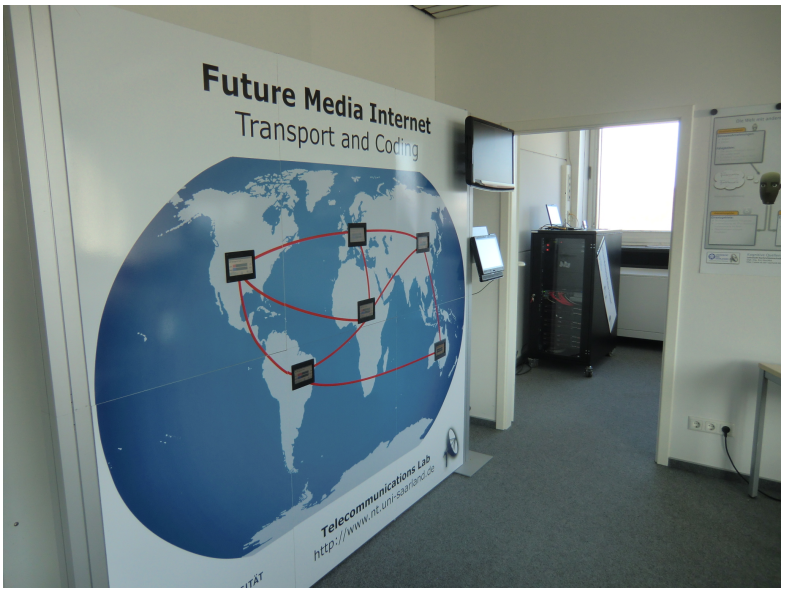

FIGURE 7.2: The LWAN demonstrator.

The LWAN demonstrator has been successfully presented at different international events and conferences, including

- NEM Summit 2011[2],
- Intel VCI Governance Board Meeting 2011[3],
- CeBIT 2012[4],
- DaaD Science Tour 2014[5].

The LWAN demonstrator consists of six network nodes, one sender, one receiver, and a central instance providing a working point for configuration and evaluation. This point is called *controller*. The network nodes are equipped with 4-port Intel network interface cards (NIC) in order to provide a non-trivial node degree. Each of these devices is individually connected to a central switch. Thus, a logical link in the demonstrator corresponds to two physically wired links. In order to separate these links VLAN tagging is used. This ensures a central connectivity by a star topology, while separating individual links. The devices are installed in a mobile network rack.

Each node can act as a relay, and thus the LWAN demonstrator implements a multi-hop transmission. The node relays are able to operate on fixed UDP and PRRT[6] connections

[2]http://www.nem-summit.eu (2015/01/29)
[3]http://www.intel-vci.uni-saarland.de (2015/01/29)
[4]http://www.cebit.de (2015/01/29)
[5]https://www.daad.org (2015/01/29)
[6]http://www.nt.uni-saarland.de/en/projects/running-projects/prrt.html (2015/01/29)

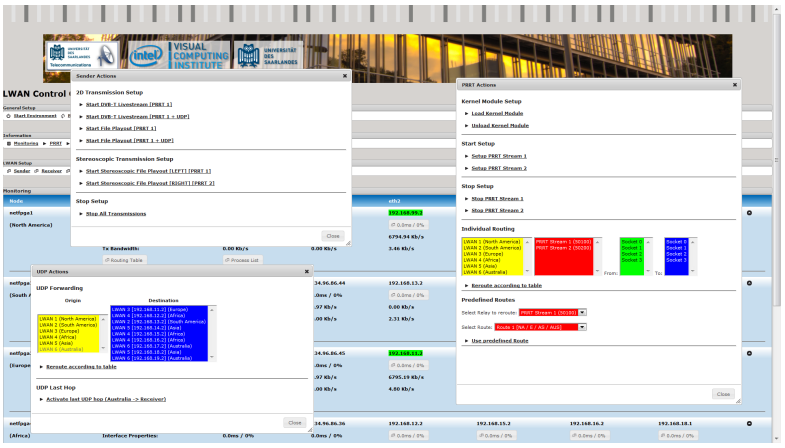

FIGURE 7.3: The LWAN control HTML interface.

and are individually implemented. Whereas UDP does not require side information for the transmission, the PRRT relays must have information about delay and residual error-rate on a link. This information must be preconfigured at each node.

The LWAN controller is the central instance that provides information to the network devices. It controls the used transmission routes in the network, the streaming source, the link characteristics, and the PRRT protocol information. It must interact with the individual components of the network. The communication channels are mainly based on chains of HTML, JavaScript, PHP, and BASH scripts. This allows a flexible access and deployment, but limits the provided functionality. Each stage in the communication chain must be individually adapted and interdependencies carefully maintained. The LWAN controller intelligence allows an adequate management of the demonstrator albeit at cost of a fixed and static scenario.

The controller additionally provides an HTML-based graphical user interface (GUI) that allows access to a large set of functionality. A part of the GUI is presented in figure 7.3.

The LWAN demonstrator represents a static proof of concept implementation of a small network testbed. It works on a fixed set of nodes and has static control mechanisms including a variety of different web-based technologies. The demonstrator consists of a valuable set of functionality for pure network research. However, it lacks options to interact with nodes in a generic and intuitive way. Despite having shown that the proposed traffic optimization schemes work in an IP network, the static structures limit the LWAN demonstrator to be a pure network simulator without flexible components. Consequently, this demonstrator provides the conceptual basis for a more sophisticated testbed based on Software Defined Networking.

7.2 Software Defined Networking Testbed

Software Defined Networks (SDN) provide a suitable and efficient platform for developing and testing new network protocols and transport paradigms. SDN represents a highly efficient networking approach with many opportunities for accessing information or packet header modifications as introduced in section 2.7. The presented testbed has been developed and is maintained at the *Telecommunications Lab*[7], in conjunction with the *Intel Visual Computing Institute*[8]. It focuses on the facilitation of research in the domain of multimedia content transport and extends the LWAN demonstrator presented in section 7.1. It contains a large set of interoperability and controlling features. Network devices and their corresponding software implementations are generally closed source and consequently impractical for research.

Established wide area networking (WAN) environments are generally not accessible for testing, as they are deployed for productive scenarios.

This SDN environment enables researchers to implement new transport ideas, such as *Loss Domain Separation (LDS)*, that has been discussed in chapter 5. New transmission paradigms and architectures can be flexibly developed and maintained in a separated and approximately fully accessible environment.

The SDN testbed has been gradually extended by external nodes to enable global collaborations in this research area. The maintenance of external nodes implies a scenario of wide area networking with conditions close to real world transport parameters and characteristics. The infrastructure can be currently used by partners located in Finland, India, and Germany. It provides a unified platform for multiple collaborations focusing on improvements for future media transmissions. The network environment is additionally used for research. It is part of a demonstrator within the *Software-Cluster* project *SINNODIUM*[9] and has also been presented to public during CeBIT 2014[10]. The infrastructure includes mechanisms to emulate special network conditions, such as packet loss or transmission delays.

More information about this *Software Defined Networking* environment is publicly available on the project website[11].

In the course of this thesis, the testbed has been continuously extended. Multiple features and modular extensions have been developed, which are presented in this chapter.

[7] http://www.nt.uni-saarland.de (2015/01/29)
[8] http://www.intel-vci.uni-saarland.de (2015/01/29)
[9] http://www.software-cluster.org/en/research/projects/joint-projects/sinnodium (2015/01/29)
[10] http://www.cebit.de (2015/01/29)
[11] http://www.openflow.uni-saarland.de (2015/01/29)

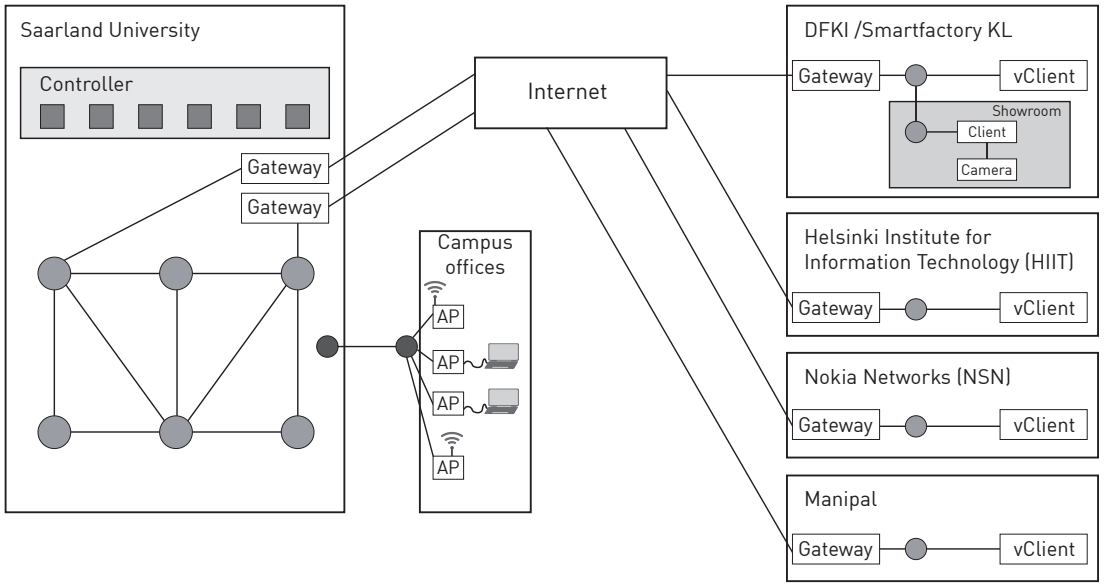

FIGURE 7.4: Topology overview of the SDN environment deployed at Saarland University.

7.2.1 Testbed Topology

The topology of the testbed comprises a mixture of virtual and physical environments as shown in figure 7.4. A *virtual* environment significantly increases flexibility and decreases hardware maintenance effort. In order to provide demonstrations focusing on research results, it is required to use real *physical* hardware, such as displays and mobile computing equipment, and wireless access. A connection between the VMware[12]-based virtual environment and the campus network infrastructure has been established by special VLAN tagging [127]. These VLANs are implemented as non-routable subnets and acting as trivial network cables. It is possible to connect a computer device to a predefined network jack on campus and being immediately connected to the virtual testbed environment. This access is realized by network sockets distributed around the university campus. The core of the network remains inside the virtual environment and is composed of bare Ubuntu[13] server machines running the OpenFlow v1.0 [22, 154] reference software implementation[14]. These virtual machines are connected via separate subnetworks that individually act as direct links.

All SDN nodes communicate with the main controller via an out-of-band (OOB) connection. This control network is exclusively used by the *OpenFlow@Saarland University* project.

[12]http://www.vmware.com (2015/01/29)
[13]http://www.ubuntu.com (2015/01/29)
[14]http://archive.openflow.org/wp/downloads/ (2015/01/29)

The main part containing all logic for the network functionality is a set of controllers. In this testbed five different controllers are maintained for different purposes, such as release, development, or for external partners. Controllers and SDN infrastructure nodes are connected via a proxy device, called *FlowVisor*[15]. This device acts as a packet dispatcher according to a specified ACL policy as described in section 7.2.1.3.

For further research a set of commodity hardware access points with an updated firmware have been deployed. Wireless access points from TP-LINK of type *WT1053* have been installed, and their operating system has been updated to *OpenWRT* [155].

While the core network resides at Saarland University, multiple external nodes are bidirectionally connected via the Internet. There are three participating research institutions: the *Helsinki Institute for Information Technology (HIIT)*[16], the *SmartFactory*[17] as part of the *German Research Center for Artificial Intelligence (DFKI)*[18], and the *University of Manipal*[19] in India. Additionally, *Nokia Solutions and Networks (NSN)*[20] decides to participate in the testbed as the first industrial partner. The connection works transparently and the SDN functionality is fully usable from any location. The external parts of the network are connected via *Gateways* and *Remote Site Nodes*, as presented in the next sections.

7.2.1.1 The WAN Gateways

A WAN gateway is particularly software running on a dedicated hardware device. It allows a packet exchange with external nodes via a multiplexed wide area IP connection carrying local network information. It contains at least three different network interface cards (NIC):

- **Gateway NIC:** This NIC is connected to the WAN and transmits UDP [2] packets containing Ethernet [156] frames of data and control packets. It constitutes the interface that connects to another gateway.
- **Control NIC:** The control NIC is connected to the OOB control network. It manages Ethernet frames of control packets interchanged between controller and SDN nodes.

[15] https://openflow.stanford.edu/display/DOCS/Flowvisor (2015/01/29)
[16] http://www.hiit.fi (2015/01/29)
[17] http://www.smartfactory.de (2015/01/29)
[18] http://www.dfki.de (2015/01/29)
[19] http://www.manipal.edu (2015/01/29)
[20] http://www.nsn.com (2015/01/29)

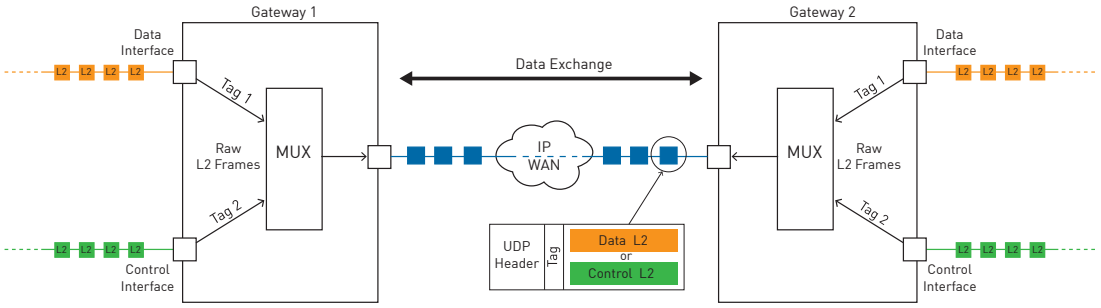

FIGURE 7.5: Basic principle of a WAN gateway used in the testbed environment.

- **Data NIC:** The data NIC is connected to the data network, that transmits information between the SDN end hosts. It handles Ethernet frames of data packets originated by the end hosts. It does not contain any SDN control data.

Figure 7.5 illustrates the basic operation principle between two WAN gateways G1 and G2. In case a packet arrives on the control or data NIC of G1, the *whole* Ethernet frame is captured and stored in the payload field of a UDP packet. A unique tag is inserted in the UDP packet to identify that this packet belongs either to the control or data network. The UDP packet is finally sent via the gateway NIC to a connected gateway G2 via an IP-based wide area network (WAN). The gateway G2 receives the UDP packet, identifies the tag, and extracts the payload. As the payload represents a full and valid Ethernet frame, it is put on its corresponding control or data NIC. Connected gateways must obviously agree on the used tags, in order to avoid misdirected packet delivery. This process ensures a transparent connection of networks, as Ethernet frames are captured and put back without any modification. Both parts of the network see the original packet information from the Ethernet layer and above, independent of their location.

The current software implementation is based on the original *capsulator*[21] application. The basic implementation only supported raw IP sockets and transmitted them over IP WANs. The recent version extended by the Telecommunications Lab uses UDP multiplexed IP connections. UDP connections ease the application of gateways in company networks with restrictive firewall ACLs. This modified version of the original *capsulator* software can be publicly found at GitHub[22].

In conjunction with the modified gateway application an intuitive web application has been developed. It enables a comfortable control and monitoring of the gateway via an HTTP-based graphical user interface. It can be found on the project website[23].

[21] http://archive.openflow.org/wk/index.php/Capsulator (2015/01/29)
[22] https://github.com/UdS-TelecommunicationsLab/Capsulator (2015/01/29)
[23] http://www.openflow.uni-saarland.de

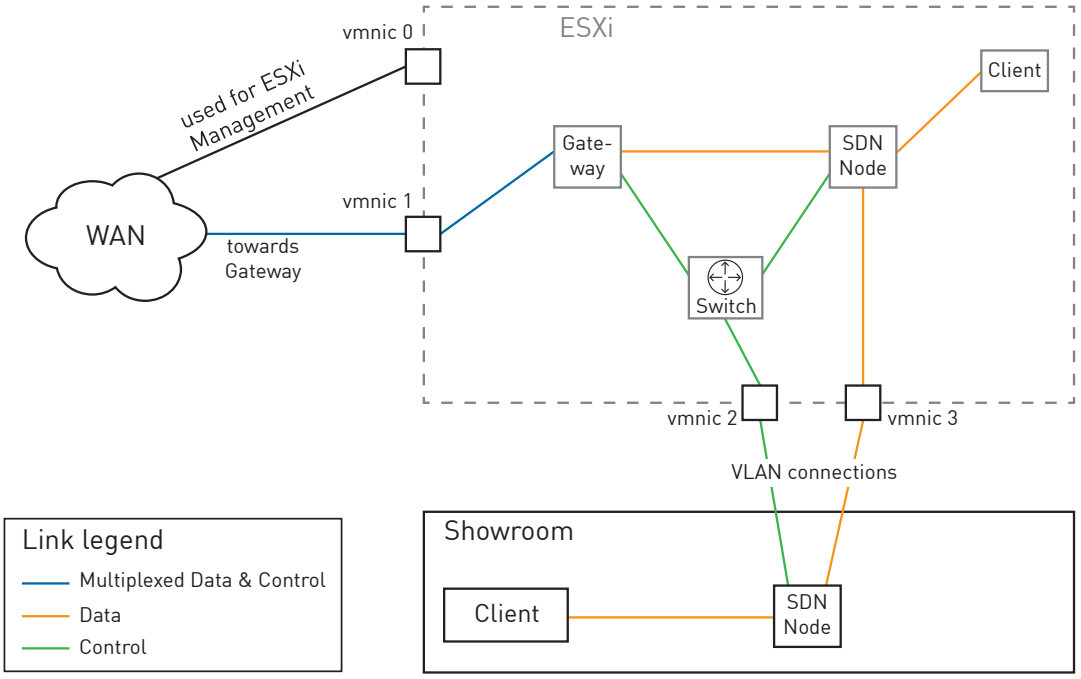

FIGURE 7.6: Schematic illustration of a remotely used ESXi SDN environment.

7.2.1.2 Remote Site SDN Nodes

During the project multiple external partners could be found that agreed with the local placement of an external node inside their network. As the core testbed network is connected to these external nodes via IP/UDP connections, these connections represent wide area links through the Internet. This allows checking transport protocols against real world phenomena as jitter, delay and packet loss. Figure 7.6 depicts the schematic setup of a remote site environment. External nodes are set up via a small virtualization environment based on a single VMware ESXi 5.1 host [157]. On this host, there are three different virtual nodes:

- **Gateway:** This gateway multiplexes the internal data and control traffic into one UDP stream that is eventually sent to another gateway as described in section 7.2.1.1. On the other side, the multiplexed stream is demultiplexed and the packets are put on the corresponding internal interfaces.
- **SDN Node:** This machine represents a standard SDN node. It runs OpenFlow 1.0 and works as any other SDN node.
- **End Device:** This device is used to simulate a receiver or source. For other end devices in the network, its location inside a virtual environment is completely transparent.

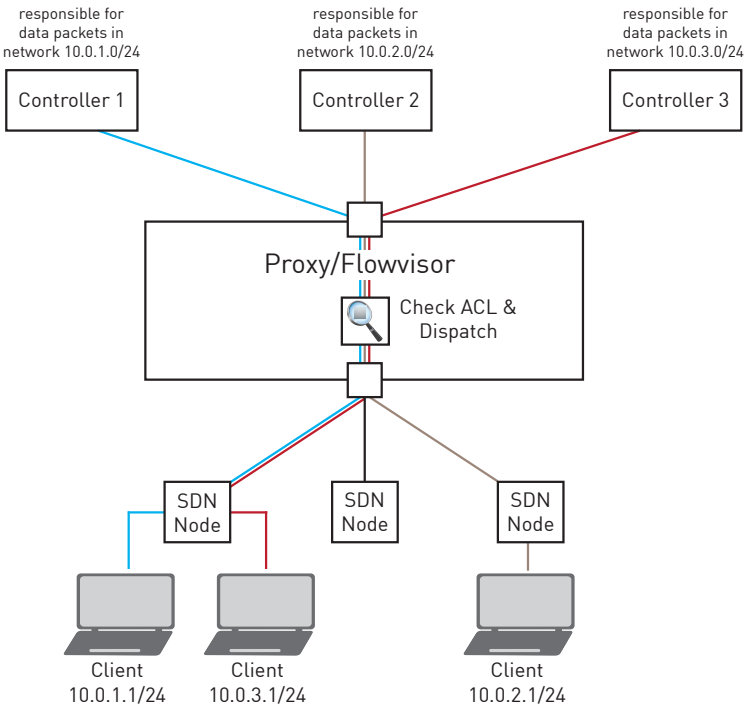

FIGURE 7.7: Connection scheme with multiple controllers.

7.2.1.3 Multiple Controllers

As an experimental testbed for research and development purposes, it is crucial to maintain a flexible basis containing multiple controllers charged with individual tasks and managed by different persons. The environment runs on OpenFlow 1.0 and it is consequently not possible to connect a single SDN node to multiple controllers. This feature has been defined in later OpenFlow specifications. Using OpenFlow 1.0 implies the application of a proxy between the SDN nodes and multiple controllers. Each node connects to the proxy instead of the controller. The proxy then represents a controller for the node, while being connected to one or more controllers. Figure 7.7 illustrates this connection scheme.

A set of access control policies enables the proxy to dispatch incoming packets to the available controllers. As an example, the proxy can assign packets to controllers according to their source MAC or IP addresses, as well as by the help of the used transport protocol or any other OSI layer 2-4 information. The proxy transparently connects controller and nodes, as neither do recognizes the presence of the proxy in between. The software running on the proxy is called *flowvisor* [158] and was initially provided by the native OpenFlow project.

7.2.1.4 Link Manipulations

Link manipulations, such as latency and packet loss probability, are required to enable a simulation of varying network conditions. In case a transport protocol is tested for its error correction capabilities, the network must be able to introduce some errors according to a certain probability distribution. All testbed SDN nodes are running an open Ubuntu[24] operation system. This allows to use the tool `netem` carried with the traffic shaper application `tc`[25]. It principally allows to add *delay*, *packet loss*, *packet jitter*, and some other characteristics to outgoing packets of a specific network interface. `Netem` utilizes the existing *Quality of Service (QoS)* [7] and *Differentiated Services (DiffServ)* [10] implementations in the Linux kernel. Both have been discussed in chapter 3.

Figure 7.8 shows the main operating principle of `tc/netem` for outgoing traffic. Data coming from a network socket is enqueued in the software traffic queue and forwarded to the hardware buffers, where it is finally sent to the network. The network emulation modifies the *queuing disciplines (qdisc)* of the software buffer in order to delay or drop packets. For an OpenFlow-based SDN node, the command looks as follows:

```
tc qdisc [add|change] ...
  dev _IFACE_ parent 1:fffe handle 101: ...
  netem delay _DELAY_ms loss _LOSS_
```

In this command `_IFACE_` represents the network interface selected for this queue manipulations, `_DELAY_` refers to the packet delay time in milliseconds, and `_LOSS_` refers to the packet loss probability emulated by the traffic shaper.

As an example, applying this command on `eth3` for introducing `45 ms delay` and `7 % packet loss`, the interface is manipulated as follows:

```
>tc qdisc show
qdisc htb 1: dev eth1 root refcnt 2 r2q 10 default fffe direct_packets_stat 0
qdisc htb 1: dev eth2 root refcnt 2 r2q 10 default fffe direct_packets_stat 0
qdisc htb 1: dev eth3 root refcnt 2 r2q 10 default fffe direct_packets_stat 0
qdisc netem 101: dev eth3 parent 1:fffe limit 1000 delay 45.0ms loss 7%
qdisc htb 1: dev eth4 root refcnt 2 r2q 10 default fffe direct_packets_stat 0
qdisc htb 1: dev eth5 root refcnt 2 r2q 10 default fffe direct_packets_stat 0
```

`tc/netem` provides many different packet queue manipulations. The presented command can be extended straightforward in order to emulate the desired network conditions.

[24] http://wwww.ubuntu.com (2015/01/29)
[25] http://man7.org/linux/man-pages/man8/tc-netem.8.html (2015/01/29)

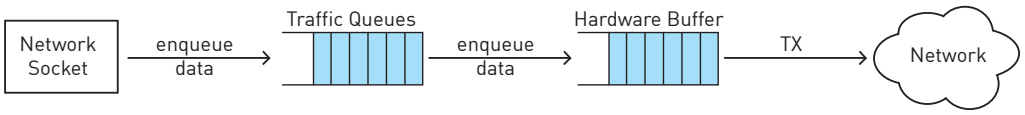

FIGURE 7.8: Basic link manipulation setup for outgoing packets.

7.2.2 Controller Design and Features

The main intelligence of a *Software Defined Network* is represented by its controller and the implemented logic. It constitutes the control plane. This SDN environment features the *NOX Classic* [159] controller. It supports rapid prototyping based on the programming language *Python* [160]. Besides NOX software, the testbed contains a JAVA-based *Floodlight* [161] controller installation. This enables a flexible and comfortable developing process as it includes a sophisticated OpenFlow 1.0 programming framework. Figure 7.9 presents a schematic overview of the general controller design and the main components, independent of the specific implementation. During initialization, the core modules register for multiple events such as `packet-in`, `datapath`, `statistics`, or `link events`. `Packet-in` events are further differentiated between individual packet types as IP [1], ARP [138], or ICMP [162] and dispatched to specific packet handler functions. This leads to a packet-sensitive and fast packet handling at a minimum of overhead and blocking. Maintenance functions handle general controller-internal administrative tasks. They are frequently invoked.

The present controller design consists of four main interacting modules residing in the main controller architecture:

- **Packet Handling:** This module handles all incoming packets. It consists of a set of specialized event handler. According to the packet type, different actions are performed and other modules incorporated. In case special functionality is desired, e.g. with multicast handling, the required intelligence is applied by the individual packet handler.

- **Data Store:** The data store provides a reliable cache for different types of information, such as connected data paths, links or clients as well as running flows or gathered statistics. It represents the core data functionality and is involved in nearly every action of the controller.

- **Event Handling:** Event handlers listen to special occurrences in the network. For example, these handlers are called in case a new data path connects to the controller, a link comes up or down, as well as when a switch sends statistics about its flow table. The communication event handler is responsible for providing

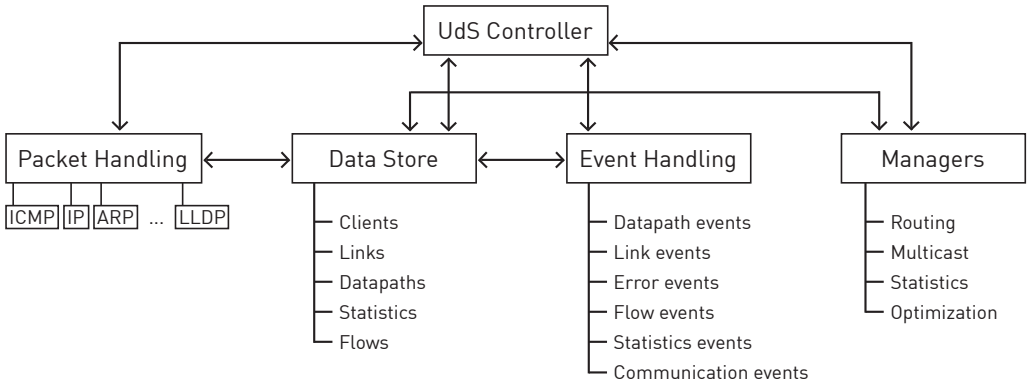

FIGURE 7.9: Overview of the controller and its modules and features.

information to the outside via a predefined API, e.g. based on *JSON* [163] file exchange. Each handler may invoke different actions from different modules.

- **Manager:** Managers contain maintenance functionality. They provide logic for routing, optimization, or multicasting as well as raw statistic evaluation. Managers access the data store and work on cached data.

The proposed design is highly flexible. New modules can be quickly announced and connected to the present modules. Replacements of modules can be performed without large overhead, e.g. in case new optimization mechanisms are required.

A subset of controller modules, that have been developed during this thesis, is presented in the next sections.

7.2.2.1 Multicast

Multicast enables *one-to-many* communication and implies a reduced network load in case multiple receivers are interested in the same content. The *Internet Group Management Protocol v3 (IGMPv3)* [164] is used to organize multicast groups. IGMP messages are exchanged between edge routers and clients. The presented multicast mechanisms represent an experimental implementation and work for all multicast addresses in the reserved block 225.0.0.0 - 231.255.255.255 as defined in [165]. An IGMPv3 framework following RFC3376 [164] has been developed to allow a standard conform IGMP message flow for applications. All IGMP messages are captured, analyzed and evaluated at the controller in order to establish appropriated traffic dissemination structures. Figure 7.10 illustrates the workflow of the multicast handling. When a multicast packet enters the packet processing chain, its content is analyzed first. In case the packet carries a multicast *report*, it is extracted and individually handled. If the packet content is *streaming*

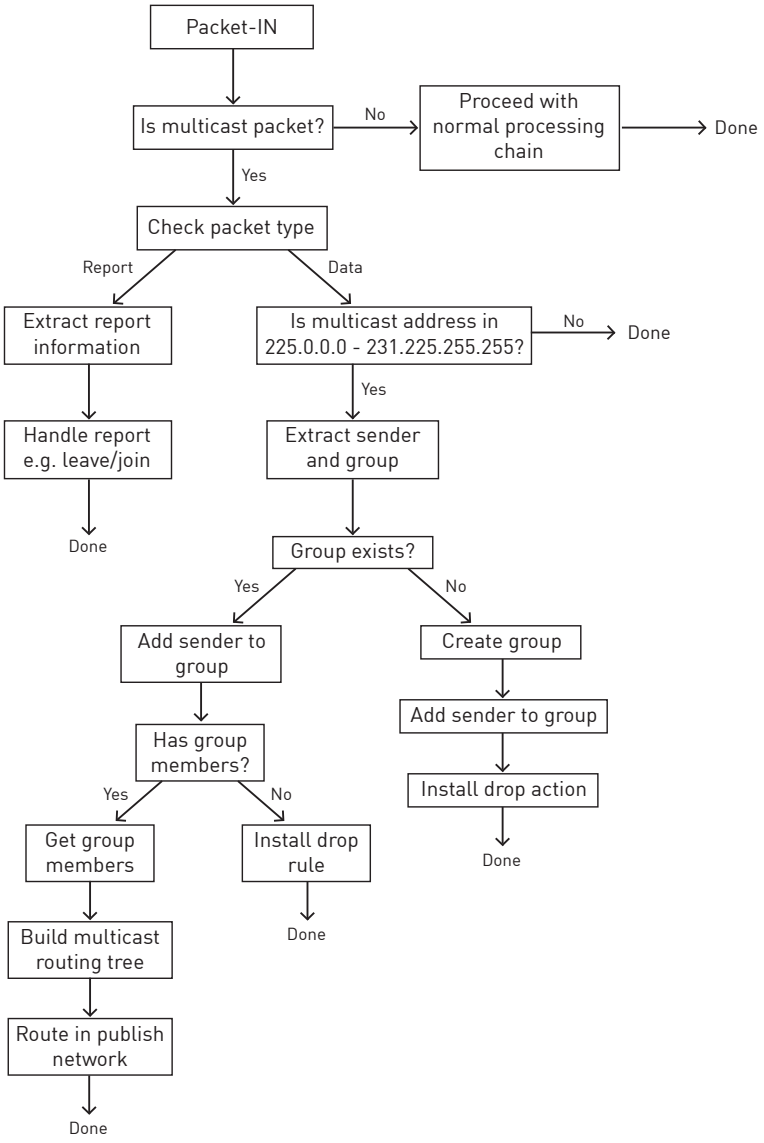

FIGURE 7.10: Multicast handling implemented in the SDN environment.

data and the present multicast address is valid, the sender address and the included group information is cached. In case the intended group does not already exist, it is created and the corresponding sender added to this group. A *packet drop rule*, associated with the corresponding multicast data flow, is eventually installed at the SDN node that is directly connected to the sender. This rule avoids flooding the network with multicast data packets with no receiver. It is replaced by a valid forwarding rule in case a receiver joins the specific multicast group.

Otherwise, if the group already exists the sender is added to it. In case the group has no assigned members, a *packet drop rule* is installed as in the previous case. If there are members, a group individual multicast routing tree is generated and finally published in

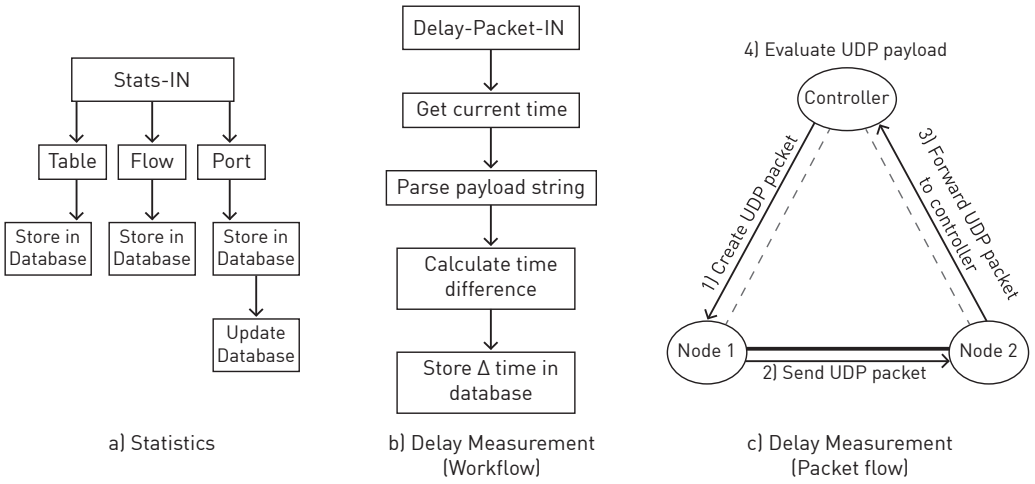

FIGURE 7.11: Schematic statistics handling as implemented in the SDN environment.

the network. The controller repeatedly sends group specific *membership queries* to all clients in order to achieve recent information about the receiving status of the clients.

7.2.2.2 Statistics

The present SDN environment runs on OpenFlow 1.0 [154]. As this OpenFlow API version does not provide any information about data rate, packet loss, or delay of an individual link between two connected SDN switches, it is required to derive this kind of parameter by available information. This information must be calculated using the number of received and sent packets (and bytes) on each port for every SDN infrastructure node. Measuring the delay of a link connecting two SDN nodes is difficult as OpenFlow does not provide any information about timestamps in the packets. Consequently, a mechanism has been developed to measure the latency between two SDN nodes via UDP packets.

Figure 7.11 presents the handling of generally specified OpenFlow statistic messages and delay measurement packets.

Part a) of figure 7.11 illustrates the handling of default `statistics-in (STATS-IN)` events. These include the events of incoming statistics packets according to [154]. In case a default `statistics-in` event is invoked, the incoming statistics are disassembled and stored in the database. These stored values are accessed by other parts of the controller, forwarded to the visualization, or used to calculate average measures. For example, *port statistics* include information about the amount of sent or received bytes and packets. This information is used to gather the current data rates and packet loss rates for specific links connecting two SDN switches. After caching port statistics, this

information is used for further processing and the database is updated. The controller itself frequently sends asynchronous `statistics request` messages to the SDN nodes. These trigger the nodes to response with a message containing all required statistics.

The *packet loss rate (PLR)* is calculated from the present port statistics. It relates the number of sent n_{tx} and received n_{rx} packets of two connected switches as shown in equation (7.1).

$$PLR(s_1, s_2) = \frac{n_{tx}(s_1) - n_{rx}(s_2)}{n_{tx}(s_1)} \tag{7.1}$$

In this equation s_1 and s_2 denote the switches connected by the link and $n_{tx|rx}$ represent the number of packets sent and received, respectively.

The data rate of a link connecting two SDN nodes can analogously be determined as shown in equation (7.2).

$$D_{tx}(s_1) = \frac{b_{tx,2}(s_1) - b_{tx,1}(s_1)}{t_2 - t_1} \tag{7.2}$$

In equation (7.2) $b_{tx,1|tx,2}$ represent the amount of sent bytes at time instances t_1 and t_2.

Delay measurement requires more interactivity with the network environment. It is implemented by individually sent UDP packets on a per-link basis. These UDP packets are initiated by the controller and carry a timestamp indicating their node departure time. Each UDP packet additionally includes information about the source and destination node and physical switch-port.

The corresponding workflow is presented in figure 7.11 part b) and the networking behavior is illustrated in figure 7.11 part c). When the UDP packet arrives at the controller again, as it has been forwarded to it by the receiving SDN node, the payload string is parsed and the included information is used to calculate the overall time difference for the specific link. A typical delay measuring payload string can be defined as:

```
SRC_SWITCH#DST_SWITCH#SRC_PORT#_DST_PORT#TIMESTAMP
```

The SDN nodes use a special set of rules to immediately forward incoming UDP delay measuring packets. This set matches packets with special MAC and IP addresses as well as UDP ports that uniquely identify these UDP packets.

As the UDP packet is created and finally parsed only at the controller, the timestamps also represent the time at the controller. Consequently, the calculated time difference also includes two different parts. The transport time between controller and SDN node, as well as certain queuing times at the SDN nodes are contained within the time difference. In case an OOB controller communication is assumed, the transport latencies

can be neglected. The queue times add a significant amount of time to the whole processing chain. Some controller frameworks, as *Floodlight*, provide special functionality to directly *flush* the created packet at the source SDN node, and do not enqueue it in the device.

The precision of this mechanism approximately approaches the real network latency conditions and can be gracefully used for network optimization.

7.2.2.3 Network Flow Optimization

In traditional network environments individual devices do not have a global view of the network. They have knowledge about adjacent nodes or a small area of the network, but their actual view is strictly limited. Switching and forwarding decisions are identified on a bounded and particular isolated set of information. The network flow optimization approach significantly improves the application of global QoS rules, as it is preferable for the transport of multimedia content. With the application of SDN environments, the controller acts as a central entity that comprises global knowledge about flow information, SDN nodes, links, clients, and statistics. The presented module exploits all information, sets up a mathematical model based on *Mixed Integer Linear Programming (MILP)* [135], and solves the model according to the constraints given by the multimedia application. The achieved results and routes are translated into SDN flow actions and distributed in the network.

In [137] a controller module called *Network Information Managed Flow Optimization (NIMFO)* has been developed. It provides an interface between the controller and an external optimization framework. The module registers for events occurring in the SDN environment such as `link up/down`, `node up/down`, or `packet_in`. The collected information is used to build and frequently update a mathematical model representing the current network topology and its conditions. An overview of the NIMFO module and the information flows are given in figure 7.12.

The model is described by the *Zuse Institute Mathematical Programming Language (ZIMPL)* [166]. It represents a simple but powerful language to describe mathematical programs. The ZIMPL notation of the model is forwarded to the *Solving Constraint Integer Program (SCIP)* [167], that finally solves the mathematical program. The results are subsequently fed back to NIMFO. It analyzes the solution and triggers adequate modifications in the network via the main controller architecture.

More details about these features and the implementation are presented in [137].

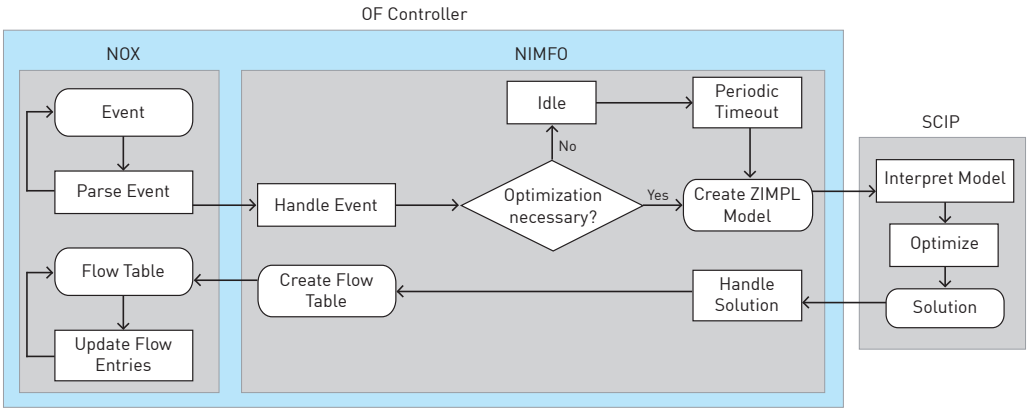

FIGURE 7.12: The network flow optimization module.

7.2.3 Specialized SDN Applications

Different applications have been developed that aim to support the application and maintenance of SDN environments. These applications do not extend the controller in terms of new intelligence for switching or routing. The main focus is on an extension that enables an intuitive and comfortable way for interaction with the network environment. The developed tools and mechanisms make use of already existing information provided by the controller. Information is evaluated and visualized to the user in a convenient appearance. The presented SDN-related applications help users and administrators of an SDN environment to obtain a better understanding of structures, processes, and events occurring in the network.

7.2.3.1 SDN Visualization

Software Defined Networks are highly interactive, adaptive, and flexible environments. They allow network engineers to expediently develop new network transmission approaches. This new kind of network technology consequently includes a large set of accessible information that is present at the controller. This information represents a valuable set of knowledge when approaching processes and mechanisms in the core of the network infrastructure. The presence of information on network events is advantageous to debug occurring issues and failures. A wide variety of users require interactive and information-rich user interfaces that allow modifying, monitoring, and triggering special interests and actions inside the network. In [168] a visualization for SDN environments has been developed that is web-based and contains flexible and interchangeable modules. It is applicable for a wide range of different user categories, such as network engineers, researchers or network administrators. The visualization architecture contains three interacting domains as illustrated in figure 7.13: *SDN*, *Backend*, and *Client*.

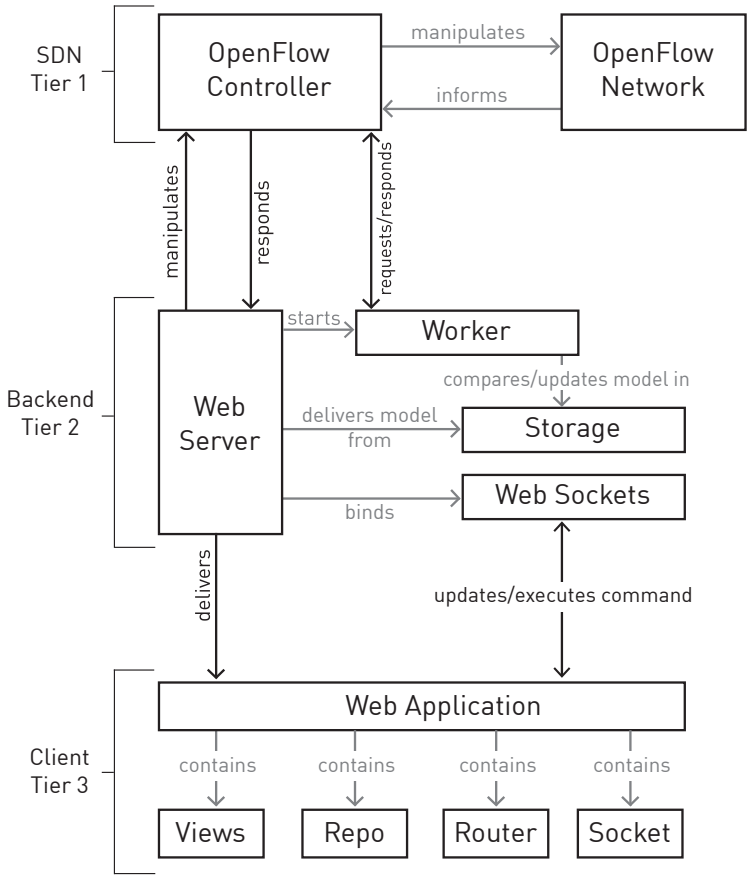

FIGURE 7.13: The SDN visualization architecture.

The *SDN* module maintains and provides the collected information about the network environment. It offers access to this information via a special interface format as specified in [168]. The module also allows to invoke functions on the SDN controller that are triggered by external events. As an example, it is possible to change the used routing metric by invoking a function located in this SDN module. The module then manages the required set up process to trigger this modification event inside the controller logic.

The *Backend* module polls information from the *SDN* module on the controller. It includes specialized logic which is used by the visualization architecture. It consists of three modules that perform different actions. The first module, i.e. the webserver and worker, performs a mapping between the arriving information from the *SDN* module into an internally used format. The second module, i.e. the storage, maintains the internal database and keeps all data up to date. The third module, i.e. the web sockets, is listening and responding incoming requests from the *Client* module.

The *Client module* communicates with the *Backend* module in real-time. It provides the main interface for users and follows the *Model-View-Controller (MVC)* principle. The

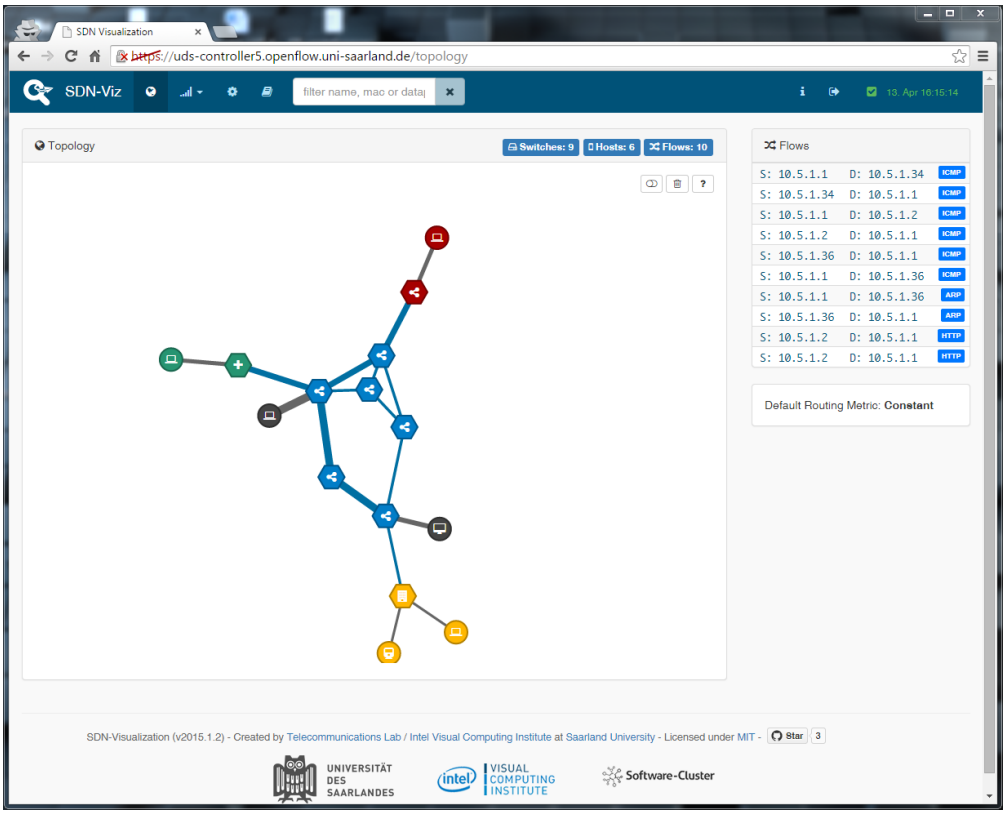

FIGURE 7.14: SDN visualization screenshot.

visualization architecture applies a compression to the required information between the modules in order to lower the required information exchange data rate. It represents a configurable application that also includes an information augmentation.

An exemplary screenshot of the visualization's main view is presented in figure 7.14.

For further details on the specification aspects of this visualization architecture refer to [168]. The presented SDN visualization is publicly available at GitHub[26].

7.2.3.2 Automatic Detection of non-SDN Devices

Software Defined Networks ease the development of new network technologies. The controller represents the central instance containing all information and intelligence for the network environment. It reliably detects SDN nodes via corresponding events, for example, as specified in the OpenFlow specification [154]. The specification does not contain a mechanism to detect non-SDN devices that are connected to the network. Non-SDN devices are user end devices, servers offering services to the SDN clients, or specialized networking devices as LDS relays that have been presented in chapter 5. A common end user device is recognized by the controller only in case that it sends

[26] https://github.com/UdS-TelecommunicationsLab/SDN-Visualization (2015/01/29)

data to the SDN environment. Then the controller identifies the SDN node and the corresponding switch port where the end user device is connected to.

The goal of this feature is to extend an existing SDN environment with a straightforward auto detection and configuration mechanism, while introducing as little additional management effort as possible. The non-SDN nodes individually advertise their roles and capabilities to the network. After these announcements have been received by the controller, the messages are analyzed and subsequent set up processes are autonomously triggered by the controller. The controller keeps full authority of the network that is additionally enriched by further information about end devices.

A feature like this is highly advantageous for a wide variety of applications such as *Content Delivery Networks (CDN)* [62] or audio-visual transmission *relaying*. For CDNs it is beneficial to obtain an immediate notification of recently connected data store devices, as mirroring and reproduction cycles can be instantaneously adjusted. This accordingly results in smaller convergence cycles. For multimedia transmissions the automatic advertisement of error coding relays would be desirable to autonomously create loss domains. The concept of *Loss Domain Separation* has been presented in chapter 5.

In [169] the automatic device detection has been implemented as an extension of the *Link Layer Discovery Protocol (LLDP)* as specified in IEEE 802.1ab [170]. It provides two modes: *automatic discovery and status monitoring* as well as *configuration* of non-OpenFlow devices. In both cases it exploits the organizational specific *Type-Length-Value (TLV)* entries that are already present in the LLDP specification [170]. Whereas automatic discovery is included in LLDP, it is proposed to use organizational specific TLVs to advertise the role of the station along with additional potentially necessary information as a metric value. A declarative approach is chosen to exchange configuration information, since LLDP is specified as a unidirectional protocol. The controller regularly advertises the required configuration to the non-SDN device that is included in an organizational specific TLV. This TLV may be distributed over one or more LLDP packets. The receiving non-SDN device parses the packet and triggers the required configuration. Acknowledgements of performed setup modifications are contained in subsequent periodically sent advertisement packets. This ensures a stateless LLDP message exchange with no special mechanisms for packet handling. Figure 7.15 presents an overview of the configuration workflow for both, controller and station.

The presented approach is called *Augmented LLDP (ALLDP)* [169]. The implementation of ALLDP results in two different software applications. The first is an extension of the controller functionality. It represents a module that is able to parse arriving ALLDP packets. In this context it defines a new packet handler as introduced in section 7.2.2.

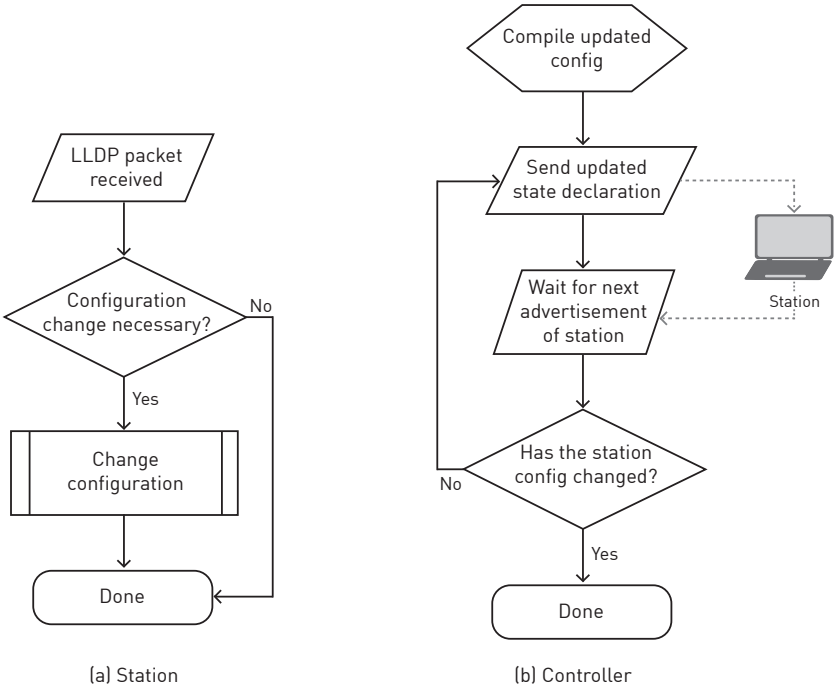

FIGURE 7.15: ALLDP workflows for non-SDN stations and the network controller.

The second software implementation resides on the non-SDN stations and periodically advertises information via ALLDP messages to the controller.

Chapter 8

Conclusion

In the last decade, network environments significantly evolved to cope with the increasing amount of data originated by audio-visual applications. However, current infrastructures fundamentally rely on approaches developed in the beginnings of the computer era. Multimedia applications introduce highly specialized demands for transport. The efficient transmission of multimedia content over wide area networks is a challenging topic. Multiple requirements must be met, such as timely delivery, low residual error rates, and network infrastructure specific characteristics.

This thesis discusses new technologies and strategies for a highly efficient transmission of multimedia content over wide area IP-based networks. As audio-visual applications introduce inherent new objectives to their data delivery, adjusting the transmission environments represents a fundamental improvement. There is an undeniable need for new strategies for exchanging multimedia information. Today's applications use end-to-end connections that neither incorporate dynamic interactions with the infrastructure nor applying multimedia aware transport mechanisms.

The goal of this thesis is to find different ways to optimize present transport mechanisms, develop fundamental new delivery strategies, and combine them for the future media Internet. In this sense, a set of evolutionary and revolutionary approaches have been discussed.

8.1 Summary

This thesis leads to the following fundamental conclusions for optimized transport of multimedia data in future wide area networks:

- The evolutionary improvement of current network infrastructures requires a new kind of routing metrics that rely on a set of different characteristics rather than a single measure.
- Interactions with network environments are advisable. This leads to a closer communication between transmission parties, applications, and network devices. The more specific both entities can exchange knowledge, the higher the eventually achieved transmission gain.
- The *spatial dimension* of network infrastructures must be considered. Especially with respect to error correction it is possible to achieve significant improvements in efficiency. This includes decreased transmission data rate, shorter latencies, and higher *Quality of Service (QoS)*.
- The spatial segmentation of a network is linked with the task of finding suitable *intersection locations*. Efficient metric definitions and more network knowledge lead to a highly efficient resource utilization, also with legacy algorithms.
- New network infrastructure technologies are required. This includes the use of evolutionary approaches that extend current infrastructures in order to ease the transmission of multimedia data and revolutionary strategies, such as *Software Defined Networks (SDN)* or *Network Function Virtualization (NFV)*. Finding a suitable and viable combination of legacy and future technologies enables a more flexible and dynamic transmission management.
- *Linear Programming (LP)* models have been proposed to represent optimum unicast, multicast, and hybrid flow distributions in network infrastructures. They can be tackled by default LP solvers and provide a fundamental basis for future optimization.

Network Metrics

Network metrics represent specific network characteristics. Metrics are used to rate network segments and decide for an appropriate set of them. Traditionally, simple metrics are used, such as the hop count between two network nodes. Complex applications, such as multimedia sources, suffer from such inflexible and static network decision criteria.

The extension of this basis to a multi-dimensional space including new characteristics is considered to be valuable with respect to latency, jitter, and individual segment error probability.

In this work network metrics have been presented and discussed that appear suitable for multimedia transmissions. As the most current network applications rely on one-dimensional network metrics, multiple approaches have been discussed to transform a set of metrics into a single metric value. The transformation enables legacy architectures to handle transmission with new content types.

Loss Domain Separation

Avoiding default end-to-end transmission approaches and establishing multi-hop connection between source and sink leads to a *Loss Domain Separation (LDS)*. Adjacent segments are linked together by special network nodes that contain logic to terminate transmissions on any OSI layer. In case this separation scheme is employed in a transparent fashion, the end devices do not recognize any differences compared to traditional end-to-end schemes as the network cares for termination and reestablishment of the connection. LDS implies an individual handling of network segments, e.g. for error correction. For example, *retransmission* schemes can be used for segments with small propagation delays whereas *Forward Error Correction (FEC)* schemes are applied to segments with larger latencies. Exploiting the network infrastructural information leads to a higher networking efficiency compared to default end-to-end schemes.

In this thesis a *Loss Domain Separation (LDS)* approach in packet-switched networks is described for a basic error correction approach without time limitations. Additionally, LDS is discussed for a retransmission and hybrid scheme with a strict time constraint. The requirements and challenges with this approach have been shown, while providing additional information about the selection of LDS intersection locations. The segmentation of network infrastructures is accompanied by the selection of suitable intersection locations. Multiple metric-based approaches have been presented that evaluate network information and provide a set of possible intersection locations inside the network.

New Network Technologies

Software Defined Networking (SDN), *Network Function Virtualization (NFV)*, and the concept of *Next Generation Networks (NGN)* lead to a new consideration of transport networks. Whereas default networks are hard to manage and adjust in real-time, these new technologies allow a smart and dynamic approach to modify the network according

to a multitude of events. They also enable a fine granular configuration on a streaming flow basis with the opportunity to transparently modify packet information. In case of new transport paradigms, such as *LDS*, these architectures allow a rapid development and deployment with central logic. Owing to the global knowledge in conjunction with flexible mechanisms, the transport of multimedia data is significantly eased.

An SDN-based environment with various specific implementations has been established. It is connected to, and used by, multiple external researchers in different countries. It represents a flexible network infrastructure for new transport approaches for numerous content types including multimedia content.

8.2 Outlook

The presented mechanisms and approaches hit various topics with high relevance for future media transmission applications. In this section potential extensions to the proposed ideas are discussed.

Network Topology Aware Resource Allocation

The spatial dimension of network infrastructures is not yet considered in an optimized and practicable way. Transmissions are configured to work in an end-to-end fashion and applications specify end-to-end demands. As shown in this thesis, the application of transport relays is an efficient solution for optimized communications, the end-to-end demands must also be usable in multi-hop environments. Different aspects have to be considered. Appropriate mechanisms are required to distribute end-to-end demands among a multi-hop network path, according to topology-related attributes as segment delay and loss. Considering the spatial characteristics in combination with an end-to-end resource allocation yields a significant improvement, compared to default network environments.

Optimization of Metric Calculations

Traditional network metrics incorporate a one-dimensional parameter space. This leads to an under-representation of important characteristics for multimedia delivery. Future network mechanisms must reflect more complex infrastructural information to achieve optimized network behavior. In order to avoid the modification of existing software and hardware implementations, a convenient mapping must be investigated, that transforms an arbitrary number of input parameter to a single value, that can be used by

already deployed instances. In principal, multiple different approaches may exist and the application scenario specifies the selection of an individual mapping function. This transformation is invoked prior to the actual legacy routing procedure and can also be applied in modern traffic engineering mechanisms. Introducing multi-dimensional metrics for routing in wide area networks represents a vital requirement and evolutionary improvement towards the future media Internet.

Routing Metrics

As shown in this thesis, the basis for routing decisions is a crucial factor. Special types of content require different network transmission characteristics that must be suitably reflected by the selected path between source and sink. It remains an important topic to find smart and practical mechanisms to provide the required information to the routing mechanisms without neglecting side information. This represents a fundamental key for optimal future media Internet transmissions with content aware character. Hence, this includes the revolutionary development of completely new architectures and algorithms as well as the evolutionary modification of already existing technologies. Metrics utilize network characteristics and express them as values. For the future media Internet it remains a crucial task to identify remarkable characteristics and parameters in networks. Data mining approaches represent a suitable first step towards this topic. Especially for new network technologies, such as *Software Defined Networks (SDN)*, interesting and challenging topics appear with this research area. As an example, the location of data mining sensor significantly influences the performance and granularity of the collected data sets. This challenge faces the whole process including preparation, pre-processing, analysis, and post-processing of network data sets.

Network Data Mining

Networks represent highly fluctuating structures with various components and parameters. Static and long term based behavior evaluation leads to a loss of valuable information including periodically congested areas, segments with recurring link or node failures, or seasonally emerging user behaviors. This evaluation consequently constitutes efficient knowledge for transport management focusing on reliable and optimized interconnections. The collection of information is based on the selection of required information and their acquisition. Selection includes the task of defining indicator parameters usable by the application environment and a suitable filtering of available information. The acquisition contains the development of aggregation instances and their location within the network. Multiple combinations of collection mode and location are possible and

must reflect the superior purpose. Collecting data from active network environments supports the establishment of reliable and viable transport connections over wide area networks in order to minimize consequences arising from otherwise hidden events.

Related Publications

[R1] Karl, Michael; Herfet, Thorsten: *Transparent Multi-Hop Protocol Termination*, 28th IEEE International Conference on Advanced Information Networking and Applications (AINA-2014), Victoria, Canada, May 2014

[R2] Karl, Michael; Gruen, Jochen; Herfet, Thorsten: *Multimedia Optimized Routing in OpenFlow Networks*, 19th IEEE International Conference on Networks (ICON 2013), Singapore, December 2013

[R3] Gruen, Jochen; Karl, Michael; Herfet, Thorsten: *Network Supported Congestion Avoidance in Software-Defined Networks*, 19th IEEE International Conference on Networks (ICON 2013), Singapore, December 2013

[R4] Karl, Michael; Polishchuk, Tatiana; Herfet, Thorsten; Gurtov, Andrei: *Mediating Traffic With Strict Delivery Constraints*, IEEE International Symposium on Multimedia, Irvine, December 2012

[R5] Polishchuk, Tatiana; Karl, Michael; Herfet, Thorsten; Gurtov, Andrei: *Scalable Architecture for Multimedia Multicast Internet Applications*, 13th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), San Francisco, June 2012

[R6] Karl, Michael; Herfet, Thorsten: *On the Efficient Segmentation of Network Links*, IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (ISBMSB), Erlangen, June 2011

[R7] Gorius, Manuel; Miroll, Jochen; Karl, Michael; Herfet, Thorsten: *Predictable Reliability and Packet Loss Domain Separation for IP Media Delivery*, IEEE International Conference on Communications (ICC), Kyoto, June 2011

[R8] Karl, Michael; Gorius, Manuel; Herfet, Thorsten: *A Distributed Multilink Media Transport Approach*, International Conference on Intelligent Network and Computing (ICINC), Kuala Lumpur, November 2010

[R9] Karl, Michael; Gorius, Manuel; Herfet, Thorsten: Routing: *Why less intelligence is sometimes more clever*, IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB), Shanghai, March 2010

[R10] Gorius, Manuel; Karl, Michael; Herfet, Thorsten: *Optimized Link-level Error Correction for Internet Media Transmission*, NEM-Summit, St. Malo, September 2009

Bibliography

[6] Jingxuan Liu and Nirwan Ansari. "Public Switched Telephone Network". In: *Handbook of Computer Networks: Key Concepts, Data Transmission, and Digital and Optical Networks, Volume 1* (2008), pp. 158–177.

[9] R. Jain. "Quality of Experience". In: *IEEE MultiMedia* 11.1 (Jan. 2004), pp. 96–95. ISSN: 1070-986X. DOI: 10.1109/MMUL.2004.10000. URL: http://dx.doi.org/10.1109/MMUL.2004.10000.

[12] Benjamin Frank et al. "Content-aware Traffic Engineering". In: *SIGMETRICS Perform. Eval. Rev.* 40.1 (June 2012), pp. 413–414. ISSN: 0163-5999. DOI: 10.1145/2318857.2254819. URL: http://doi.acm.org/10.1145/2318857.2254819.

[13] Ning Wang et al. "An overview of routing optimization for internet traffic engineering". In: *Communications Surveys & Tutorials, IEEE* 10.1 (2008), pp. 36–56.

[14] Christina Büsing. *Graphen- und Netzwerkoptimierung*. Springer Science & Business, 2010.

[16] Thomas H. Cormen et al. *Introduction to Algorithms*. 2nd. McGraw-Hill Higher Education, 2001. ISBN: 0070131511.

[17] Dingzhu Du and Xiaodong Hu. *Steiner tree problems in computer communication networks*. World Scientific, 2008.

[21] Thomas D. Nadeau and Ken Gray. *SDN: Software Defined Networks*. " O'Reilly Media, Inc.", 2013.

[22] Nick McKeown et al. "OpenFlow: enabling innovation in campus networks". In: *ACM SIGCOMM Computer Communication Review* 38.2 (2008), pp. 69–74.

[25] Mauricio G.C. Resende and Panos Pardalos. *Handbook of Optimization in Telecommunications*. Springer Science & Business Media, 2008.

[26] Dmitry Kachan, Eduard Siemens, and Vyacheslav Shuvalov. “Comparison of Contemporary Solutions for High Speed Data Transport on WAN 10 Gbit/s Connections”. In: *ICNS 2013, The Ninth International Conference on Networking and Services*. 2013, pp. 34–43.

[27] Thomas Stockhammer. “Dynamic adaptive streaming over HTTP–: standards and design principles”. In: *Proceedings of the second annual ACM conference on Multimedia systems*. ACM. 2011, pp. 133–144.

[35] John G. Proakis and Masoud Salehi. *Communication Systems Engineering*. Second. Upper Saddle River, NJ, USA: Prentice-Hall, Aug. 2001. ISBN: 0130617938.

[40] Larry L. Peterson and Bruce S. Davie. *Computer Networks: A Systems Approach*. Elsevier, 2007.

[47] Jerome H. Saltzer, David P. Reed, and David D. Clark. “End-to-End Arguments in System Design”. In: *ACM Transactions on Computer Systems (TOCS)* 2.4 (1984), pp. 277–288.

[48] Ralf Steinmetz and Klaus Wehrle, eds. *Peer-to-Peer Systems and Applications*. Vol. 3485. Lecture Notes in Computer Science. Springer, 2005. ISBN: 3-540-29192-X.

[51] Tore Opsahl, Filip Agneessens, and John Skvoretz. “Node centrality in weighted networks: Generalizing degree and shortest paths”. In: *Social Networks* 32.3 (2010), pp. 245 –251. ISSN: 0378-8733. DOI: http://dx.doi.org/10.1016/j.socnet.2010.03.006. URL: http://www.sciencedirect.com/science/article/pii/S0378873310000183.

[52] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley & Sons, 2012.

[53] James D. McCabe. *Network Analysis, Architecture, and Design*. Morgan Kaufmann, 2010.

[54] Richard W. Hamming. “Error detecting and error correcting codes”. In: *Bell System technical journal* 29.2 (1950), pp. 147–160.

[57] Gabriel Robins and Alexander Zelikovsky. “Improved Steiner tree approximation in graphs”. In: *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics. 2000, pp. 770–779.

[62] George Pallis and Athena Vakali. “Insight and Perspectives for Content Delivery Networks”. In: *Communications of the ACM* 49.1 (2006), pp. 101–106.

[64] Anurag Kumar, D Manjunath, and Joy Kuri. *Communication Networking: An Analytical Approach*. Elsevier, 2004.

[65] Markus Hufschmid. “Reed-Solomon Codes”. In: *Information und Kommunikation: Grundlagen und Verfahren der Informationsübertragung* (2007), pp. 77–86.

[72] Gonzalo Camarillo and Miguel-Angel Garcia-Martin. *The 3G IP multimedia subsystem (IMS): merging the Internet and the cellular worlds*. John Wiley & Sons, 2007.

[81] Bur Goode. “Voice over Internet Protocol (VoIP)”. In: *Proceedings of the IEEE* 90.9 (2002), pp. 1495–1517.

[94] Steve Grady. *The Book on FTTX: From Design to Deployment: a Practical Guide to FTTX Infrastructure*. ADC Telecommunications, Incorporated, 2005.

[120] Sally Floyd and Van Jacobson. “Random early detection gateways for congestion avoidance”. In: *Networking, IEEE/ACM Transactions on* 1.4 (1993), pp. 397–413.

[130] Amin Shokrollahi. “Raptor Codes”. In: *Information Theory, IEEE Transactions on* 52.6 (2006), pp. 2551–2567.

[132] Thomas Müller-Gronbach, Erich Novak, and Klaus Ritter. *Monte Carlo Algorithmen*. Springer, 2012, pp. 1–12.

[134] Bernhard Korte and Jens Vygen. *Combinatorial Optimization*. Springer-Verlag, Berlin, 2008.

[135] Taieb Mellouli and Leena Suhl. *Optimierungssysteme*. Springer-Verlag Berlin Heidelberg, 2009.

[136] Richard M. Karp. *Reducibility among combinatorial problems*. Springer, 1972.

[139] Peter Tittmann. *Graphentheorie*. Hanser Fachbuchverlag, 2003, pp. 3–446.

[140] Boris V. Cherkassky and Andrew V. Goldberg. “Negative-Cycle Detection Algorithms”. In: *Mathematical Programming* 85.2 (1999), pp. 277–311.

[141] Hans De Neve and Piet Van Mieghem. “TAMCRA: A Tunable Accuracy Multiple Constraints Routing Algorithm”. In: *Computer Communications* 23.7 (2000), pp. 667–679.

[144] I. N. Bronstein et al. *Taschenbuch der Mathematik*. 6. Frankfurt am Main: Verlag Harri Deutsch, 2005. ISBN: 3-8171-2006-0.

[145] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Vol. 3. JHU Press, 2012.

[148] Manuel Gorius and Thorsten Herfet. “Echtzeit Fehlerschutz für die drahtlose paketbasierte Fernsehübertragung”. In: *23. Annual Conference of the FKTG (Fernseh– und Kinotechnische Gesellschaft)* (2008).

[149] Urs Hoelzle. “Openflow@Google”. In: *Open Networking Summit* 17 (2012).

[152] S.S. Haykin and M. Moher. *Modern wireless communications*. Pearson Prentice Hall, 2005. ISBN: 9780130224729.

[153] Paolo Toth and Silvano Martello. *Knapsack Problems: Algorithms and Computer Implementations*. Wiley, 1990.

[155] Florian Fainelli. "The OpenWrt embedded development framework". In: (2008).

[158] Rob Sherwood et al. "Flowvisor: A network virtualization layer". In: *OpenFlow Switch Consortium, Tech. Rep* (2009).

[160] Magnus Lie Hetland. *Beginning Python*. Springer, 2005.

[167] Tobias Achterberg. "SCIP: Solving Constraint Integer Programs". In: *Mathematical Programming Computation* 1.1 (2009), pp. 1–41.

Theses

[31] Soamdev Acharya. "Techniques for improving Multimedia Communication over Wide Area Networks". PhD thesis. Cornell University, 1999.

[36] Guoping Tan. "Optimum Hybrid Error Correction Scheme under Strict Delay Constraints". PhD thesis. Saarland University, 2009.

[66] Manuel Gorius. "Adaptive Delay-constrained Internet Media Transport". PhD thesis. Saarland University, 2012.

[137] Nicolas Seiwert. "Flow Optimization for concurrent Multimedia Traffic in Software Defined Networks". MA thesis. Saarland University, Telecommunications Lab, 2014.

[166] Thorsten Koch. "Rapid Mathematical Programming". PhD thesis. Technische Universität Berlin, 2004.

[168] Andreas Schmidt. "Interactive Visualization of Software Defined Networks". Bachelor's Thesis. Saarland University, Telecommunications Lab, 2013.

[169] Frank Wassmuth. "Auto Detection and Configuration of PRRT Coding Relays in OpenFlow Networks". Bachelor's Thesis. Saarland University, Telecommunications Lab, 2013.

Standards

[8] ITU. *Advanced video coding for generic audiovisual services (ITU-T Recommendation H.264)*. International Telecommunications Union, Feb. 2014.

[23] ETSI. *Network Functions Virtualisation (NFV); Use Cases (ETSI GS NFV 001 V1.1.1)*. European Telecommunications Standards Institute. Oct. 2013.

[24] ETSI. *Network Functions Virtualisation (NFV); Architectural Framework (ETSI GS NFV 002 V1.1.1)*. European Telecommunications Standards Institute. Oct. 2013.

[30] 3GPP. *Transparent end-to-end Packet-switched Streaming Service (PSS); Progressive Download and Dynamic Adaptive Streaming over HTTP (3GP-DASH), Release 13*. 3rd Generation Partnership Project. Dec. 2014.

[38] ITU. *Information technology - Open Systems Interconnection - Basic Reference Model: The basic model (ITU-T Recommendation X.200)*. International Telecommunications Union, July 1994.

[39] ETSI. *Telecommunications and Internet converged Services and Protocols for Advanced Networking (TISPAN); NGN Functional Architecture (ETSI ES 282 001 V3.4.1)*. European Telecommunications Standards Institute. Sept. 2009.

[41] ITU. *Information technology - Open Systems Interconnection - Presentation service definition (ITU-T Recommendation X.216)*. International Telecommunications Union, July 1994.

[42] ITU. *Information technology - Open Systems Interconnection - Session service definition (ITU-T Recommendation X.215)*. International Telecommunications Union, Nov. 1995.

[43] ITU. *Information technology - Open Systems Interconnection -Transport service definition (ITU-T Recommendation X.214)*. International Telecommunications Union, Nov. 1995.

[44] ITU. *Information technology - Open Systems Interconnection - Network service definition (ITU-T Recommendation X.213)*. International Telecommunications Union, Oct. 2001.

[45] ITU. *Information technology - Open Systems Interconnection - Data Link service definition (ITU-T Recommendation X.212)*. International Telecommunications Union, Nov. 1995.

[46] ITU. *Interface between Data Terminal Equipment and Data Circuit-terminating Equipment for synchronous operation on public data networks (ITU-T Recommendation X.21)*. International Telecommunications Union, Sept. 1992.

[67] ITU. *General overview of NGN (ITU-T Recommendation Y.2001)*. International Telecommunications Union, Dec. 2004.

[68] ITU. *General principles and general reference model for Next Generation Networks (ITU-T Recommendation Y.2011)*. International Telecommunications Union, Oct. 2004.

[69] ITU. *Functional architecture of connectionless layer networks (ITU-T Recommendation G.809)*. International Telecommunications Union, Mar. 1991.

[70] ITU. *Generic functional architecture of transport networks (ITU-T Recommendation G.805)*. International Telecommunications Union, Mar. 2000.

[71] ITU. *General architectural model for interworking (ITU-T Recommendation Y.1251)*. International Telecommunications Union, Aug. 2002.

[80] ITU. *A digital modem and analogue modem pair for use on the Public Switched Telephone Network (PSTN) at data signalling rates of up to 56 000 bit/s downstream and up to 33 600 bit/s upstream (ITU-T Recommendation V.90)*. International Telecommunications Union, Sept. 1998.

[82] ITU. *Packet-based multimedia communications systems (ITU-T Recommendation H.323)*. International Telecommunications Union, Sept. 2012.

[85] ITU. *One-way transmission time (ITU-T Recommendation G.114)*. International Telecommunications Union, May 2003.

[87] ITU. *Coding of speech at 8 kbit/s using conjugate-structure algebraic-code-excited linear prediction (CS-ACELP) (ITU-T Recommendation H.225.0)*. International Telecommunications Union, June 2012.

[91] ITU. *Asymmetric digital subscriber line (ADSL) transceivers (ITU-T Recommendation G.992.1)*. International Telecommunications Union, June 1999.

[92] ITU. *Asymmetric digital subscriber line transceivers 2 (ADSL2) (ITU-T Recommendation G.992.3)*. International Telecommunications Union, Apr. 2009.

[93] ITU. *Single-pair high-speed digital subscriber line (SHDSL) transceivers (ITU-T Recommendation G.991.2)*. International Telecommunications Union, Dec. 2003.

[95] ISO. *Information technology-Dynamic adaptive streaming over HTTP (DASH)-Part 1: Media presentation description and segment formats (ISO/IEC 23009-1)*. International Organization for Standardization, 2014.

[109] ITU. *Call signalling protocols and media stream packetization for packet-based multimedia communication systems (ITU-T Recommendation H.225.0)*. International Telecommunications Union, Dec. 2009.

[110] ITU. *Control protocol for multimedia communication (ITU-T Recommendation H.245)*. International Telecommunications Union, May 2011.

[121] ITU. *Network performance objectives for IP-based services (ITU-T Recommendation Y.1541)*. International Telecommunications Union, Dec. 2011.

[122] ETSI. *Digital Video Broadcasting (DVB); Transport of MPEG-2 TS Based DVB Services over IP Based Networks (ETSI TS 102 034 V1.4.1)*. European Telecommunications Standards Institute. Aug. 2009.

[123] ETSI. *Telecommunications and Internet converged Services and Protocols for Advanced Networking (TISPAN); IPTV Architecture; IPTV functions supported by the IMS subsystem (ETSI TS 182 027 V3.5.1)*. European Telecommunications Standards Institute. Mar. 2011.

[124] ETSI. *Telecommunications and Internet converged Services and Protocols for Advanced Networking (TISPAN); NGN integrated IPTV subsystem Architecture (ETSI TS 182 028 V3.3.1)*. European Telecommunications Standards Institute. Oct. 2009.

[125] DVB. *DVB-IP Phase 1.3 in the context of ETSI TISPAN NGN (DVB Document A128)*. The Digital Video Broadcasting Project. Sept. 2008.

[127] IEEE Computer Society. *IEEE Standard for Local and metropolitan area networks - Media Access Control (MAC) Bridges and Virtual Bridge Local Area Networks (IEEE Std. 802.1Q-2011)*. Institute of Electrical and Electronics Engineers. Aug. 2011.

[128] SMPTE. *Forward Error Correction for Real-Time Video/Audio Transport Over IP Networks (SMPTE ST 2022-1:2007)*. Society of Motion Picture and Television Engineers, 2007.

[129] SMPTE. *Unidirectional Transport of Constant Bit Rate MPEG-2 Transport Streams on IP Networks (SMPTE ST 2022-2:2007)*. Society of Motion Picture and Television Engineers, 2007.

[133] IEEE Computer Society. *IEEE Standard for Local and metropolitan area networks - Link Aggregation (IEEE Std. 802.1AX-2008)*. Institute of Electrical and Electronics Engineers. Nov. 2008.

[156] IEEE Computer Society. *IEEE Standard for Ethernet (IEEE Std. 802.3-2012)*. Institute of Electrical and Electronics Engineers. Dec. 2012.

[170] IEEE Computer Society. *IEEE Standard for Local and metropolitan area networks - Station and Media Access Control Connectivity Discovery (IEEE Std. 802.1AB-2009)*. Institute of Electrical and Electronics Engineers. Sept. 2009.

Request for Comments (RFC)

[1] J. Postel. *Internet Protocol*. RFC 791 (INTERNET STANDARD). Updated by RFCs 1349, 2474, 6864. Internet Engineering Task Force, Sept. 1981. URL: http://www.ietf.org/rfc/rfc791.txt.

[2] J. Postel. *User Datagram Protocol*. RFC 768 (INTERNET STANDARD). Internet Engineering Task Force, Aug. 1980. URL: http://www.ietf.org/rfc/rfc768.txt.

[3] J. Postel. *Transmission Control Protocol*. RFC 793 (INTERNET STANDARD). Updated by RFCs 1122, 3168, 6093, 6528. Internet Engineering Task Force, Sept. 1981. URL: http://www.ietf.org/rfc/rfc793.txt.

[4] P.V. Mockapetris. *Domain names - concepts and facilities*. RFC 1034 (INTERNET STANDARD). Updated by RFCs 1101, 1183, 1348, 1876, 1982, 2065, 2181, 2308, 2535, 4033, 4034, 4035, 4343, 4035, 4592, 5936. Internet Engineering Task Force, Nov. 1987. URL: http://www.ietf.org/rfc/rfc1034.txt.

[5] P.V. Mockapetris. *Domain names - implementation and specification*. RFC 1035 (INTERNET STANDARD). Updated by RFCs 1101, 1183, 1348, 1876, 1982, 1995, 1996, 2065, 2136, 2181, 2137, 2308, 2535, 2673, 2845, 3425, 3658, 4033, 4034, 4035, 4343, 5936, 5966, 6604. Internet Engineering Task Force, Nov. 1987. URL: http://www.ietf.org/rfc/rfc1035.txt.

[7] S. Shenker, C. Partridge, and R. Guerin. *Specification of Guaranteed Quality of Service*. RFC 2212 (Proposed Standard). Internet Engineering Task Force, Sept. 1997. URL: http://www.ietf.org/rfc/rfc2212.txt.

[10] S. Blake et al. *An Architecture for Differentiated Services*. RFC 2475 (Informational). Updated by RFC 3260. Internet Engineering Task Force, Dec. 1998. URL: http://www.ietf.org/rfc/rfc2475.txt.

[11] R. Braden, D. Clark, and S. Shenker. *Integrated Services in the Internet Architecture: an Overview*. RFC 1633 (Informational). Internet Engineering Task Force, June 1994. URL: http://www.ietf.org/rfc/rfc1633.txt.

[15] J. Moy. *OSPF Version 2*. RFC 2328 (INTERNET STANDARD). Updated by RFCs 5709, 6549, 6845, 6860. Internet Engineering Task Force, Apr. 1998. URL: http://www.ietf.org/rfc/rfc2328.txt.

[18] E. Rosen, A. Viswanathan, and R. Callon. *Multiprotocol Label Switching Architecture*. RFC 3031 (Proposed Standard). Updated by RFCs 6178, 6790. Internet Engineering Task Force, Jan. 2001. URL: http://www.ietf.org/rfc/rfc3031.txt.

[19] D. Awduche et al. *RSVP-TE: Extensions to RSVP for LSP Tunnels*. RFC 3209 (Proposed Standard). Updated by RFCs 3936, 4420, 4874, 5151, 5420, 5711, 6780, 6790, 7274. Internet Engineering Task Force, Dec. 2001. URL: http://www.ietf.org/rfc/rfc3209.txt.

[20] D. Awduche et al. *Requirements for Traffic Engineering Over MPLS*. RFC 2702 (Informational). Internet Engineering Task Force, Sept. 1999. URL: http://www.ietf.org/rfc/rfc2702.txt.

[28] H. Schulzrinne et al. *RTP: A Transport Protocol for Real-Time Applications*. RFC 3550 (INTERNET STANDARD). Updated by RFCs 5506, 5761, 6051, 6222, 7022, 7160, 7164. Internet Engineering Task Force, July 2003. URL: http://www.ietf.org/rfc/rfc3550.txt.

[29] R. Fielding et al. *Hypertext Transfer Protocol – HTTP/1.1*. RFC 2616 (Draft Standard). Obsoleted by RFCs 7230, 7231, 7232, 7233, 7234, 7235, updated by RFCs 2817, 5785, 6266, 6585. Internet Engineering Task Force, June 1999. URL: http://www.ietf.org/rfc/rfc2616.txt.

[32] J. Border et al. *Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations*. RFC 3135 (Informational). Internet Engineering Task Force, June 2001. URL: http://www.ietf.org/rfc/rfc3135.txt.

[34] G. Fairhurst and L. Wood. *Advice to link designers on link Automatic Repeat reQuest (ARQ)*. RFC 3366 (Best Current Practice). Internet Engineering Task Force, Aug. 2002. URL: http://www.ietf.org/rfc/rfc3366.txt.

[37] B. Carpenter. *Architectural Principles of the Internet*. RFC 1958 (Informational). Updated by RFC 3439. Internet Engineering Task Force, June 1996. URL: http://www.ietf.org/rfc/rfc1958.txt.

[49] Y. Rekhter, T. Li, and S. Hares. *A Border Gateway Protocol 4 (BGP-4)*. RFC 4271 (Draft Standard). Updated by RFCs 6286, 6608, 6793. Internet Engineering Task Force, Jan. 2006. URL: http://www.ietf.org/rfc/rfc4271.txt.

[50] D. Oran. *OSI IS-IS Intra-domain Routing Protocol*. RFC 1142 (Historic). Obsoleted by RFC 7142. Internet Engineering Task Force, Feb. 1990. URL: http://www.ietf.org/rfc/rfc1142.txt.

[55] M. Laubach. *Classical IP and ARP over ATM*. RFC 1577 (Proposed Standard). Obsoleted by RFC 2225. Internet Engineering Task Force, Jan. 1994. URL: http://www.ietf.org/rfc/rfc1577.txt.

[56] G. Malkin. *RIP Version 2*. RFC 2453 (INTERNET STANDARD). Updated by RFC 4822. Internet Engineering Task Force, Nov. 1998. URL: http://www.ietf.org/rfc/rfc2453.txt.

[58] D. Waitzman, C. Partridge, and S.E. Deering. *Distance Vector Multicast Routing Protocol*. RFC 1075 (Experimental). Internet Engineering Task Force, Nov. 1988. URL: http://www.ietf.org/rfc/rfc1075.txt.

[59] J. Moy. *Multicast Extensions to OSPF*. RFC 1584 (Historic). Internet Engineering Task Force, Mar. 1994. URL: http://www.ietf.org/rfc/rfc1584.txt.

[60] A. Adams, J. Nicholas, and W. Siadak. *Protocol Independent Multicast - Dense Mode (PIM-DM): Protocol Specification (Revised)*. RFC 3973 (Experimental). Internet Engineering Task Force, Jan. 2005. URL: http://www.ietf.org/rfc/rfc3973.txt.

[61] B. Fenner et al. *Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)*. RFC 4601 (Proposed Standard). Updated by RFCs 5059, 5796, 6226. Internet Engineering Task Force, Aug. 2006. URL: http://www.ietf.org/rfc/rfc4601.txt.

[63] C. Partridge, T. Mendez, and W. Milliken. *Host Anycasting Service*. RFC 1546 (Informational). Internet Engineering Task Force, Nov. 1993. URL: http://www.ietf.org/rfc/rfc1546.txt.

[74] R. Droms. *Dynamic Host Configuration Protocol*. RFC 2131 (Draft Standard). Updated by RFCs 3396, 4361, 5494, 6842. Internet Engineering Task Force, Mar. 1997. URL: http://www.ietf.org/rfc/rfc2131.txt.

[75] D. Crocker. *STANDARD FOR THE FORMAT OF ARPA INTERNET TEXT MESSAGES*. RFC 822 (INTERNET STANDARD). Obsoleted by RFC 2822, updated by RFCs 1123, 2156, 1327, 1138, 1148. Internet Engineering Task Force, Aug. 1982. URL: http://www.ietf.org/rfc/rfc822.txt.

[76] P. Resnick. *Internet Message Format*. RFC 2822 (Proposed Standard). Obsoleted by RFC 5322, updated by RFCs 5335, 5336. Internet Engineering Task Force, Apr. 2001. URL: http://www.ietf.org/rfc/rfc2822.txt.

[77] J. Postel and J. Reynolds. *File Transfer Protocol*. RFC 959 (INTERNET STANDARD). Updated by RFCs 2228, 2640, 2773, 3659, 5797, 7151. Internet Engineering Task Force, Oct. 1985. URL: http://www.ietf.org/rfc/rfc959.txt.

[78] J. Postel and J.K. Reynolds. *Telnet Protocol Specification*. RFC 854 (INTERNET STANDARD). Updated by RFC 5198. Internet Engineering Task Force, May 1983. URL: http://www.ietf.org/rfc/rfc854.txt.

[83] J. Rosenberg et al. *SIP: Session Initiation Protocol*. RFC 3261 (Proposed Standard). Updated by RFCs 3265, 3853, 4320, 4916, 5393, 5621, 5626, 5630, 5922, 5954, 6026, 6141, 6665, 6878. Internet Engineering Task Force, June 2002. URL: http://www.ietf.org/rfc/rfc3261.txt.

[96] L. Ong and J. Yoakum. *An Introduction to the Stream Control Transmission Protocol (SCTP)*. RFC 3286 (Informational). Internet Engineering Task Force, May 2002. URL: http://www.ietf.org/rfc/rfc3286.txt.

[97] R. Stewart. *Stream Control Transmission Protocol*. RFC 4960 (Proposed Standard). Updated by RFCs 6096, 6335, 7053. Internet Engineering Task Force, Sept. 2007. URL: http://www.ietf.org/rfc/rfc4960.txt.

[98] E. Kohler, M. Handley, and S. Floyd. *Datagram Congestion Control Protocol (DCCP)*. RFC 4340 (Proposed Standard). Updated by RFCs 5595, 5596, 6335, 6773. Internet Engineering Task Force, Mar. 2006. URL: http://www.ietf.org/rfc/rfc4340.txt.

[99] L. Delgrossi and L. Berger. *Internet Stream Protocol Version 2 (ST2) Protocol Specification - Version ST2+*. RFC 1819 (Experimental). Internet Engineering Task Force, Aug. 1995. URL: http://www.ietf.org/rfc/rfc1819.txt.

[100] V. Jacobson, R. Braden, and D. Borman. *TCP Extensions for High Performance*. RFC 1323 (Proposed Standard). Internet Engineering Task Force, May 1992. URL: http://www.ietf.org/rfc/rfc1323.txt.

[101] J. Klensin. *Simple Mail Transfer Protocol*. RFC 5321 (Draft Standard). Internet Engineering Task Force, Oct. 2008. URL: http://www.ietf.org/rfc/rfc5321.txt.

[102] J. Myers and M. Rose. *Post Office Protocol - Version 3*. RFC 1939 (INTERNET STANDARD). Updated by RFCs 1957, 2449, 6186. Internet Engineering Task Force, May 1996. URL: http://www.ietf.org/rfc/rfc1939.txt.

[103] A.K. Bhushan. *File Transfer Protocol*. RFC 354. Obsoleted by RFC 542, updated by RFCs 385, 454, 683. Internet Engineering Task Force, July 1972. URL: http://www.ietf.org/rfc/rfc354.txt.

[104] K. Lougheed and Y. Rekhter. *Border Gateway Protocol (BGP)*. RFC 1163 (Historic). Obsoleted by RFC 1267. Internet Engineering Task Force, June 1990. URL: http://www.ietf.org/rfc/rfc1163.txt.

[105] E. Rescorla. *HTTP Over TLS*. RFC 2818 (Informational). Updated by RFCs 5785, 7230. Internet Engineering Task Force, May 2000. URL: http://www.ietf.org/rfc/rfc2818.txt.

[106] Y. Goland et al. *HTTP Extensions for Distributed Authoring – WEBDAV*. RFC 2518 (Proposed Standard). Obsoleted by RFC 4918. Internet Engineering Task Force, Feb. 1999. URL: http://www.ietf.org/rfc/rfc2518.txt.

[107] H. Schulzrinne and S. Casner. *RTP Profile for Audio and Video Conferences with Minimal Control*. RFC 3551 (INTERNET STANDARD). Updated by RFCs 5761, 7007. Internet Engineering Task Force, July 2003. URL: http://www.ietf.org/rfc/rfc3551.txt.

[108] H. Schulzrinne, A. Rao, and R. Lanphier. *Real Time Streaming Protocol (RTSP)*. RFC 2326 (Proposed Standard). Internet Engineering Task Force, Apr. 1998. URL: http://www.ietf.org/rfc/rfc2326.txt.

[111] K. Nichols et al. *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*. RFC 2474 (Proposed Standard). Updated by RFCs 3168, 3260. Internet Engineering Task Force, Dec. 1998. URL: http://www.ietf.org/rfc/rfc2474.txt.

[112] J. Wroclawski. *Specification of the Controlled-Load Network Element Service*. RFC 2211 (Proposed Standard). Internet Engineering Task Force, Sept. 1997. URL: http://www.ietf.org/rfc/rfc2211.txt.

[113] R. Braden et al. *Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification*. RFC 2205 (Proposed Standard). Updated by RFCs 2750, 3936, 4495, 5946, 6437, 6780. Internet Engineering Task Force, Sept. 1997. URL: http://www.ietf.org/rfc/rfc2205.txt.

[114] J. Hawkinson and T. Bates. *Guidelines for creation, selection, and registration of an Autonomous System (AS)*. RFC 1930 (Best Current Practice). Updated by RFCs 6996, 7300. Internet Engineering Task Force, Mar. 1996. URL: http://www.ietf.org/rfc/rfc1930.txt.

[115] F. Le Faucheur et al. *Multi-Protocol Label Switching (MPLS) Support of Differentiated Services*. RFC 3270 (Proposed Standard). Updated by RFC 5462. Internet Engineering Task Force, May 2002. URL: http://www.ietf.org/rfc/rfc3270.txt.

[116] B. Davie et al. *An Expedited Forwarding PHB (Per-Hop Behavior)*. RFC 3246 (Proposed Standard). Internet Engineering Task Force, Mar. 2002. URL: http://www.ietf.org/rfc/rfc3246.txt.

[117] F. Baker, J. Polk, and M. Dolly. *A Differentiated Services Code Point (DSCP) for Capacity-Admitted Traffic*. RFC 5865 (Proposed Standard). Internet Engineering Task Force, May 2010. URL: http://www.ietf.org/rfc/rfc5865.txt.

[118] J. Heinanen et al. *Assured Forwarding PHB Group*. RFC 2597 (Proposed Standard). Updated by RFC 3260. Internet Engineering Task Force, June 1999. URL: http://www.ietf.org/rfc/rfc2597.txt.

[119] D. Grossman. *New Terminology and Clarifications for Diffserv*. RFC 3260 (Informational). Internet Engineering Task Force, Apr. 2002. URL: http://www.ietf.org/rfc/rfc3260.txt.

[126] J. Babiarz, K. Chan, and F. Baker. *Configuration Guidelines for DiffServ Service Classes*. RFC 4594 (Informational). Updated by RFC 5865. Internet Engineering Task Force, Aug. 2006. URL: http://www.ietf.org/rfc/rfc4594.txt.

[131] M. Watson, M. Luby, and L. Vicisano. *Forward Error Correction (FEC) Building Block*. RFC 5052 (Proposed Standard). Internet Engineering Task Force, Aug. 2007. URL: http://www.ietf.org/rfc/rfc5052.txt.

[138] D. Plummer. *Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware*. RFC 826 (INTERNET STANDARD). Updated by RFCs 5227, 5494. Internet Engineering Task Force, Nov. 1982. URL: http://www.ietf.org/rfc/rfc826.txt.

[142] JP. Vasseur et al. *Routing Metrics Used for Path Calculation in Low-Power and Lossy Networks*. RFC 6551 (Proposed Standard). Internet Engineering Task Force, Mar. 2012. URL: http://www.ietf.org/rfc/rfc6551.txt.

[143] T. Winter et al. *RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks*. RFC 6550 (Proposed Standard). Internet Engineering Task Force, Mar. 2012. URL: http://www.ietf.org/rfc/rfc6550.txt.

[147] M. Handley et al. *TCP Friendly Rate Control (TFRC): Protocol Specification*. RFC 3448 (Proposed Standard). Obsoleted by RFC 5348. Internet Engineering Task Force, Jan. 2003. URL: http://www.ietf.org/rfc/rfc3448.txt.

[162] J. Postel. *Internet Control Message Protocol*. RFC 792 (INTERNET STANDARD). Updated by RFCs 950, 4884, 6633, 6918. Internet Engineering Task Force, Sept. 1981. URL: http://www.ietf.org/rfc/rfc792.txt.

[163] D. Crockford. *The application/json Media Type for JavaScript Object Notation (JSON)*. RFC 4627 (Informational). Obsoleted by RFC 7159. Internet Engineering Task Force, July 2006. URL: http://www.ietf.org/rfc/rfc4627.txt.

[164] B. Cain et al. *Internet Group Management Protocol, Version 3*. RFC 3376 (Proposed Standard). Updated by RFC 4604. Internet Engineering Task Force, Oct. 2002. URL: http://www.ietf.org/rfc/rfc3376.txt.

[165] M. Cotton, L. Vegoda, and D. Meyer. *IANA Guidelines for IPv4 Multicast Address Assignments*. RFC 5771 (Best Current Practice). Internet Engineering Task Force, Mar. 2010. URL: http://www.ietf.org/rfc/rfc5771.txt.

Web References

[33] J. Neale et al. *Enhancements for TCP performance enhancing proxies*. US Patent 6,975,647. Dec. 2005. URL: http://www.google.co.in/patents/US6975647.

[73] Open Networking Foundation. *OpenFlow Switch Specification (Version 1.3.0).* Available at https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.0.pdf (2014/11/13).

[79] Cisco Systems. *Visual Networking Index.* Available at http://www.cisco.com/c/en/us/solutions/service-provider/visual-networking-index-vni/index.html (2014/11/13).

[84] Cisco Systems. *Frame Relay.* Available at http://docwiki.cisco.com/wiki/Frame_Relay (2014/11/13).

[86] QWest Communications Corporation. *QWest Internet Network Service Level Agreement.* Available at http://www.centurylink.com/legal/docs/Qwest_Internet_SLA__10_07_04_.pdf (2014/11/13).

[88] Cisco Systems. *Quality of Service for Voice over IP.* Available at http://www.cisco.com/c/en/us/td/docs/ios/solutions_docs/qos_solutions/QoSVoIP/QoSVoIP.pdf (2014/12/08).

[89] Cisco Systems. *Configuring Voice over Frame Relay.* Available at http://www.cisco.com/c/en/us/td/docs/ios/12_2/voice/configuration/guide/fvvfax_c/vvfvofr.html (2014/11/13).

[90] Cisco Systems. *Configuring Voice over ATM.* Available at http://www.cisco.com/c/en/us/td/docs/ios/12_2/voice/configuration/guide/fvvfax_c/vvfvoatm.html (2014/11/13).

[146] Cisco Systems. *Cisco Unified Communications System Release 8.x SRND.* Available at http://www.cisco.com/c/en/us/td/docs/voice_ip_comm/cucm/srnd/8x/uc8x.pdf (2014/11/27).

[150] Juniper Networks. *When is the right time to deploy SDN?* Available at https://www.juniper.net/us/en/dm/sdn/ (2014/11/12).

[151] Cisco Systems. *Software Defined Networking (SDN).* Available at http://www.cisco.com/web/solutions/trends/sdn/index.html (2014/11/27).

[154] Open Networking Foundation. *OpenFlow Switch Specification (Version 1.0.0).* Available at https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.0.0.pdf (2014/11/13).

[157] VMware, Inc. *VMware ESXi.* Available at http://www.vmware.com/products/esxi-and-esx/overview.html (2014/11/13).

[159] *NOX*.
Available at http://www.noxrepo.org (2014/11/13).

[161] Project Floodlight (Supported by Big Switch Networks). *Project Floodlight*.
Available at http://www.projectfloodlight.org/floodlight/ (2014/11/13).