TECHNISCHE UNIVERSITÄT MÜNCHEN Lehrstuhl für Kommunikationsnetze Fachgebiet Medientechnik

Video Multicast in Peer-to-Peer **Networks**

Francisco de Asís López Fuentes

Vollständiger Abdruck der von der Fakultät Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender: 2.

Univ.-Prof. Dr.-Ing. Klaus Diepold Prüfer der Dissertation: 1. Univ.-Prof. Dr.-Ing. Eckehard Steinbach apl. Prof. Dr.-Ing. habil. Walter Stechele

Die Dissertation wurde am 05.08.2008 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 07.01.2009 angenommen.

Video Multicast in Peer-to-Peer Networks

Francisco de Asís López Fuentes

January 31, 2009

To Iliana, Ingrid and Magnolia

Acknowledgments

I am very grateful to my advisor Prof. Dr.-Ing. Eckehard Steinbach who guided my research and shared his ideas and insights during these years. I appreciate his time, support, and advices given by my work. This doctoral thesis would never have been possible without his support, guidance and professional attitude.

I would also like to thank Prof. Dr.-Ing. Walter Stechele for accepting to be the second examiner of this thesis and Prof. Dr.-Ing. Klaus Diepold for heading the committee.

During my stay at the Technische Universität München (TUM) I had the pleasure to interact with the members of the Media Technology Group (MTG) and Lehrstuhl für Kommunikationsnetze (LKN). A research group with diversity and high motivation. I want to thank to all them. Especially, the collaboration and discussions with Shoaib Khan, Wei Tu, Dr.-Ing. Ingo Bauermann and Peter Hinterseer was always very fruitful. I would like to thank Dr.-Ing. Martin Maier, for all the technical support at LKN.

This research was partially supported by the Consejo Nacional de Ciencia y Tecnologia (CONACyT) of Mexico since October 1, 2003 until March 31, 2008, and by the Deutscher Akademischer Austausch Dienst (DAAD) since April 1, 2008 until September 30, 2008. A German course was sponsored by DAAD between April 1, 2003 to September 30, 2003.

Gracias a mi familia y amigos por su apoyo y comprensión en todo momento. A mi padres por la formación que ellos me dieron. A mi esposa Iliana por toda su paciencia durante el desarrollo de este trabajo y a mis hijas Ingrid y Magnolia Nicole por toda felicidad que ellas nos han traido. A mis hermanos, suegros y cuñados por su motivación y apoyo.

Munich, July 2008

Francisco de Asís López Fuentes

Abstract

Peer-to-peer (P2P) video streaming is a promising solution to enable efficient video distribution over the Internet. The nodes or peers in this type of system can take the role of both a server and of a client at the same time. During video distribution each peer acts as a relay node contributing it upload capacity. This thesis focuses on a specific topic known as P2P Video Multicast within the P2P streaming area. The contribution of this thesis is to present network-adaptive techniques and novel architectures which improve the video delivery in terms of end-to-end delay, scalability and video quality for different scenarios. The proposed architectures are supported by an analytical framework, extensive simulations and evaluations in real-world scenarios using the PlanetLab overlay testbed.

Kurzfassung

Peer-to-Peer (P2P) Video Streaming ist ein vielversprechender Ansatz, um eine effiziente Verteilung von Videos über das Internet zu ermöglichen. Die beteiligten Netzknoten können gleichzeitig die Rolle eines Servers oder eines Clients übernehmen. Während der Videoübertragung agiert jeder Peer als Relais-Knoten und trägt seine Upload-Kapazität bei. Diese Arbeit beschäftigt sich mit P2P Video Multicast, welches einen Teilbereich von P2P Video-Streaming darstellt. Der Hauptbeitrag der Arbeit besteht in neuen Architekturen und adaptiven Methoden, die die Videoübertragung hinsichtlich Ende-zu-Ende-Verzögernung, Skalierbarkeit und Videoqualität in verschiedenen Szenarien verbessert. Die vorgeschlagenen Architekturen werden sowohl analytisch, durch umfangreiche Simulationen, sowie durch eine Implementierung im weltumspannenden Overlay-Netz PlanetLab evaluiert und validiert.

Contents

List of Figures xii			xiii
List of Tables xvi			xvii
Ab	brev	iations and Acronyms	xix
No	otatio	n	xxi
1.	Intro 1.1. 1.2. 1.3. 1.4.	Dduction Motivation for Media Delivery on Peer-to-Peer Networks Contributions Organization Performance Metrics	1 1 2 4 4
2.	Bac 2.1. 2.2. 2.3. 2.4. 2.5. 2.6. 2.7.	kground in Media DeliveryMedia RepresentationMedia Compression2.2.1.H.264/AVC Video Coding2.2.2.Scalable Video Coding2.2.3.H.264/SVC Scalable Video Coding ExtensionDigital Video FormatsMedia ApplicationsMedia Delivery2.5.1.Communication Modes2.5.2.Media Delivery Modes2.5.3.Protocols for Media StreamingDelivery Infrastructures2.6.1.IP-Multicast2.6.2.Content Delivery Network2.6.3.Application Layer Multicast (ALM)2.6.4.P2P NetworksChapter Summary	 7 8 9 9 10 11 12 12 12 13 14 15 15 16 22
3.	Stat 3.1. 3.2. 3.3.	e of the Art in Overlay-based Distribution Introduction 3.1.1. Tree-based P2P Streaming 3.1.2. Forest-based Overlay 3.1.3. Mesh-based Overlay 3.1.3. Mesh-based Overlay for Media Streaming 3.2.1. ZigZag Application of Forest-based Overlays for Media Streaming 3.2.1. ZigZag	 23 24 25 26 27 27

		3.3.1. SplitStream	28 28
		333 M-ary Trees	20
	34	Application of Mesh-based Overlays for Media Streaming	30
	0.1.	3.4.1 Mutualcast	30
	35	Hybrid Tree / Mosh Overlay	35
	3.5. 3.6	Multi Source Multicast	35
	3.0. 2.7	A Clobal Tagthade Planet ab	26
	5.7.	271 Mutual asst Implementation and Evaluation	26
	20	S.7.1. Mutualcast implementation and Evaluation	20
	3.8. 2.0	Chamber Guerran area	3/
	3.9.	Chapter Summary	38
4.	Hier	archical Collaborative Multicast	39
	4.1.	Introduction	39
	4.2.	Description	40
	4.3.	Cluster Organization	42
		4.3.1. Cluster Formation	42
		4.3.2. Flow Control Mechanism	44
		4.3.3. Caching Mechanism	44
		4.3.4. Redistribution Mechanism between Clusters	46
	4.4.	Simulation	49
	4.5.	Implementation	50
	4.6.	Evaluation	52
	4.7.	Chapter Summary	56
5.	Mult	i-Source Video Multicast	59
	5.1.	Description	61
	5.2.	Throughput-based Analysis	62
		5.2.1. Sources with Joint Rate Allocation for Different Rate Streams	62
		5.2.2. Sources with Joint Rate Allocation for Same Rate Streams	65
		5.2.3. Sources with Independent Rate Allocation for Same Rate Streams	68
		5.2.4. Sources with Independent Rate Allocation for Different Rate Streams .	71
	5.3.	PSNR-based Analysis	72
		5.3.1. Sources with Joint Rate Allocation for Different Video Quality Streams	73
		5.3.2. Sources with Joint Rate Allocation for Same Video Quality Streams	74
		5.3.3. Sources with Independent Rate Allocation for Same Rate Streams	77
		5.3.4. Sources with Independent Rate Allocation for Different Video Quality	
		Streams	78
	5.4.	Simulation	79
		5.4.1. Throughput-based Simulation	79
		5.4.2. PSNR-based Simulation	82
	5.5.	Implementation and Evaluation	85
		5.5.1. Implementation	85
		5.5.2. Evaluation	87
	5.6.	Chapter Summary	90
6	Ma	ry Trac based Video Multicast	02
υ.	ivi=di	ין ווכב-שמשכט אוטכט ואוטונוטמשנ	33

6. M-ary Tree-based Video Multicast

	6.1.	Motivation	94
	6.2.	System Architecture	95
		6.2.1. Building a Collection of m-ary Trees	97
		6.2.2. Example	98
	6.3.	Simulation Results	101
	6.4.	Planet Implementation and Evaluation	103
		6.4.1. Case 1: Throughput among All Participating Peers is Similar	105
		6.4.2. Case 2: Delivery Delay when the Throughput between the Source and	
		a Subsets of Peers is Highly Heterogeneous	106
		6.4.3. Case 3: Delivery Delay when the Throughput between the Subsets of	
		Peers is Weak	107
	6.5.	Chapter Summary	109
7.	Con	clusions and Future Work	111
	7.1.	Conclusions	111
	7.2.	Limitations	113
	7.3.	Future Work	113
Α.	App	endix	117
	A.1.	Simulation for Multi-source video Multicast	117
		A.1.1. Results for Throughput-based Simulation	117
		A.1.2. Results for PSNR-based Simulation	119
	A.2.	Experiments based on Scalable Video Coding	121
Bil	oliog	raphy	125

Contents

List of Figures

2.1.	Different distribution schemes: a). Unicast, b). Broadcast, c). Multicast	12
2.2.	IP Multicast	14
2.3.	Application Layer Multicast	16
2.4.	P2P Overlay Architecture	17
2.5.	A comparison among the different unstructured P2P architectures. a). Cen-	
	tralized P2P, b). Pure P2P, c). Hybrid P2P	18
3.1.	A comparison between: a) tree-based overlay, b) forest-based overlay, and c) mesh-based overlay	24
3.2	Example of an m-ary tree for 12 nodes	29
3.3	Example of a complete m-ary tree for $m = 3$	30
3.4	Mutualcast content distribution network	31
3.5	Participating peers in the Mutualcast scheme	32
3.6	Mutualcast example and its translation into a linear program for throughput	02
0.0.	maximization	34
3.7.	Mutualcast throughput measurements: a) throughput of the source, b)- d)	01
0111	throughput of the requesting peers	37
4.1	Hierarchical Collaborative Multicast architecture for 15 peers organized in	
1.1.	clusters of size 3	41
4.2.	Zone organization during cluster formation, a). Proximity constraints are not	
	considered, b). Concentric rings are used	43
4.3.	Flow control mechanism	45
4.4.	Storage blocks	46
4.5.	Flow diagram for the storage process	47
4.6.	The operation flow of the forward link on requesting peer.	48
4.7.	An example of the hierarchical collaborative multicast scheme for six request-	
	ing peers together with the corresponds linear program. The objective is to	
	maximize the overall throughput	49
4.8.	Overall throughput comparison for Mutualcast and Hierarchical Collabora-	
	tive Multicast	50
4.9.	Establishment of peer connections and cluster formation from the source'	
	perspective.	52
4.10.	Establishment of connections on the requesting peer.	53
4.11.	Packet distance comparison between Mutualcast and Hierarchical Collabora-	
	tive Multicast	53
4.12.	Organization of Planetlab nodes using Hierarchical Collaborative Multicast .	54
4.13.	Dynamic load distribution in clusters	55
4.14.	Dynamic load distribution in Mutualcast	56

4.15.	Maximum delay comparison between Mutualcast and Hierarchical Collabo- rative Multicast	56
5.1. 5.2. 5.3.	Proposed for multi-source multicast approach $\ldots \ldots \ldots \ldots \ldots \ldots$ A multi-source example with two sources and N_1 requesting peers $\ldots \ldots$ Exhaustion of ΔS_1 and ΔS_2	61 63 66
5.4.	An example of two sources with independent rate allocation and same rate streams	69
5.5.	Enforcement of the same video quality for two different videos using scalable	
5.6. 5.7.	Adaptive rate control scheme for two sources	75 76
5.8.	throughput	80 81
5.9.	The first scenario is translated into a linear program for the maximization of the aggregate video quality	83
5.10.	a). Overall <i>PSNR</i> comparison for all scenarios, b). Individual <i>PSNR</i> com-	00
5.11.	Arrangement of PlanetLab nodes during the evaluation of multi-source video	85
5.12.	multicast	88
5.13.	an adaptive mechanism for joint rate allocation	89
	Joint rate allocation using an adaptive mechanism.	89
5.14.	Rate allocation percentage for each source: a). Sources with independent rate allocation, b). Sources using an adaptive mechanism for joint rate allocation.	90
5.15.	Throughput comparison for the case of identical videos using adaptive multi- source multicast. a). Sources using an adaptive mechanism for joint rate allo- cation, b). Sources with independent rate allocation.	91
6.1.	Peers are located in several geographical locations during a multicast session	94
6.2. 6.3.	Mutualcast with same distance <i>d</i> (e.g., RTT) among peers	95
6.4	approach when block X_5 from the source is delivered	96 07
6.4. 6.5.	Preliminary m-ary tree collection	97 99
6.6.	Reduced m-ary trees collection	100
6.7.	A reduced m-ary tree collection is translated into a linear program for through-	
6.0	put maximization	101
6.8. 6.0	Overall throughput comparison	102
0.9.	a) Maximum delay b) Average delay	103
6.10	Delivery rate on each distribution tree	103
6.11.	PlanetLab nodes used in the USA and Europe to evaluate the proximity-aware	200
	collaborative multicast scheme and the Mutualcast scheme	104

6.12.	m-ary tree collection when all participating peers present a similar throughput	105
6.13.	Delivery delay to distribute all blocks when the throughput among all peers	
	is similar. The source is located at Rice University	106
6.14.	m-ary tree collection when the throughput between the source and a subset of	
	peers is highly heterogeneous	107
6.15.	Delivery delay to distribute all blocks to all peers when the throughput be-	
	tween the source and a subset of peers is highly heterogeneous. The source is	
	located at the MIT	107
6.16.	m-ary tree collection to evaluate the first experiment when throughput among	
	subsets of peers is weak. The source is located at the MIT	108
6.17.	Delivery delays obtained from the first experiment when throughput among	
	subsets of peers is weak. Source is located at the MIT	108
6.18.	m-ary tree collection to evaluate the second experiment when throughput	
	among subsets of peers is weak. The source is located at the MIT	109
6.19.	Delivery delays obtained from the second experiment when throughput among	
	subsets of peers is weak. Source is located at the MIT	110
A 1	Evenuels misture of the Methon and Developtor video converses	100
A.1.	Example picture of the Foreman video sequence.	122
A.2.	Example picture of the Foreman video sequence.	123

List of Figures

List of Tables

3.1.	The overall throughput Θ of Mutual cast $\ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots$	34
5.1.	PSNR and rate for video sequences	82
6.1.	Upload capacity C_i of the requesting peers and their proximity to the source. the two right main columns show the normalized values (compare (6.1) and (6.2))	99
A.1.	Overall throughput Θ for sources with joint rate allocation and different rate	
۸ D	Streams	117
A.2.	streams	118
A.3.	Overall throughput Θ for sources with independent rate allocation and same	110
	rate streams	118
A.4.	Overall throughput Θ for sources with independent rate allocation and differ-	110
• -		119
A.5.	PSINK for sources with joint rate allocation and different video quality streams	119
A.6.	PSINK for sources with joint rate allocation and same rate streams	120
A.7.	PSNK for sources with joint rate allocation after the PSNK enforcement	120
A.8.	PSNR for independent sources with same rate streams	121
A.9.	PSNR for independent sources with different video quality streams	121
A.10	.PSNR and rate for Mother & Daughter sequence (1 BL and 1 EL)	122
A.11	.PSNR and rate for Foreman sequence (1 BL and 1 EL)	122
A.12	.PSNR and rate for Foreman sequence (1 BL and 2 EL)	123

List of Tables

Abbreviations and Acronyms

ALM Application Level Multicast **AVC** Advanced Video Coding **CDN** Content Delivery Networks **CIF** Common Intermediate Format (352x288 pixels) **CSCW** Computer-Supported Cooperative Work **CCIR** International Radio Consultative Committee **DHT** Distributed Hash Table **DSL** Digital Subscriber Line **QCIF** Quarter Common Intermediate Format (176x144 pixels) **JSVM** Joint Scalable Video Model **IP** Internet Protocol **ISP** Internet Service Provider **ITU** International Telecommunications Union **HDTV** High-Definition Television **HTTP** Hypertext Transfer Protocol H.264/AVC Advanced Video Coding LAN Local Area Network **LP** Linear Programming **MDC** Multiple Description Coding **MPDU** Message Protocol Data Unit **MSE** Mean Squared Error **MPEG** Moving Pictures Expert Group NS2 Network Simulator 2 **NP** No Polinomial **QoS** Quality of Service

P2P Peer-to-Peer
PL PlanetLab
PSNR Peak Signal-to-Noise Ratio
RD Rate-Distortion
RTT Round Trip Time
TCP Transmission Control Protocol
UDP User Datagram Protocol
SVC Scalable Video Coding
VoIP Voice over IP
VCL Video Coding Layer
WLAN Wireless Local Area Network
WAN Wide Area Network

Notation

- N total number of vertices/nodes/peers in the considered overlay network
- N_1 total number of requesting peers in the considered overlay network
- \mathcal{N}_2 total number of helper peers in the considered overlay network
- B_R amount of data that is sent from the source to N_1 requesting peers for redistribution
- B_H amount of data that is sent from the source to N_2 helper peers for redistribution
- B_D amount of data that is directly sent from the source to N_1 requesting peers
- C_{R_i} upload capacity of requesting peer R_i
- C_{H_i} upload capacity of source H_i
- C_{S_i} upload capacity of source S_i
- Θ overall throughput in the considered system
- M number of sources in a multi-source system
- m number of child peers in an m-ary tree or cluster
- $\Delta_{S_i}\,$ residual upload capacity of source S_i
- R coding rate of a video sequence
- PSNR video distortion in dB
- C_j^n normalized upload capacity for peer R_j
- $D_i^n(j)$ normalized distance between peers R_i and R_j

Notation

1. Introduction

For many years, video has been present as an important media of entertainment and communication in the society. The recent advances of the Internet and computing technologies have opened up new opportunities to multimedia applications, where video delivery and streaming over the Internet has gained significant popularity. Currently, video playback from on-line video or news site has become part of the daily life of most Internet users. This fact has generated a technological and social revolution in media distribution and consumption.

Despite recent advances, video streaming over the Internet still presents many challenges, mainly due to the nature of the Internet, which is based on the best-effort packet transfer concept where quality-of-service (QoS) is not provided and throughput and delay fluctuations can easily lead to video playback interruptions. Media streaming systems are distinct from file-sharing systems, in which a client has to download the entire file before using it. In a media streaming session, the receiver can already consume the file while downloading. This kind of multimedia application requires high data rate, low-latency and low packet lost rate, which represent a significant challenge for the design of future network architectures.

1.1. Motivation for Media Delivery on Peer-to-Peer Networks

During the past years, streaming video over the Internet has received great attention from academia and industry due to the growing demand of the users for video services. To this end, several multimedia applications have been developed and different distribution architectures have been proposed. However, most media streaming applications are based on the traditional client-server model which leads to limitations on the achievable performance. Even the large streaming servers are not able to feed more than a few hundred streaming sessions simultaneously. The streaming server is the single point of failure and the required high rate network access is costly. In addition, the selection of the best streaming server during a session is difficult.

Since the Napster advent in 1999, Peer-to-Peer (P2P) networks have experienced a tremendous growth in popularity. This kind of networks has generated great interest in the research community who find in these systems a fast and efficient way to deliver movies, music or software files. In a P2P system, the users interact directly as a way to exchange their resources and services through the Internet [7]. P2P systems maintain their independence of the underlying physical network by using an overlay topology. In contrast to the clientserver model, in a P2P system each computer or peer can take the role of both a server and of a client. P2P networks have several advantages over the traditional client-server approach, due to their inherent scalability, distributed delivery process and flexibility, which facilitate a broad spectrum of multimedia applications.

P2P overlay networks have become a promising solution for video streaming and many P2P multimedia systems have been subsequently deployed, which include distributed file

sharing, content distribution or multimedia sensor networks. However, P2P media streaming still has many open issues that must be solved to achieve high and constant video quality, as well as low initial latencies. To this end, new P2P media streaming applications are being investigated and developed. This thesis explores one of these applications known as P2P Video Multicast.

P2P Multicast is a solution that allows for the distribution of content to an audience without the need for any special support from the network [10], and where the upload capacity of the participating peers is only considered to forward the content. P2P Multicast uses a delivery mechanism to distribute media content to the participating peers, which are self-organized into an overlay structure. The overlay structure is based on a single tree, multiple or on meshes, while the delivery protocol is used to manage the media streaming to each peer via this overlay structure. Although, P2P Multicast is a promising technology to content distribution in scenarios such as videoconferencing or IP-based TV, until now its potential has been greatly limited by low video quality as a result of low throughput to the receivers.

The purpose of this thesis is to provide media streaming architectures which help to reduce end-to-end delay, increase the scalability and enhance video quality. This work studies P2P multicast for two different distribution problems, namely one-to-many content distribution and many-to-many content distribution. This thesis investigates the benefits of proximity, rate allocation and scalable video encoding to improve the performance of a P2P multicast system. In particular, proximity issues are used to deal with the low latency video streaming problem. Rate allocation strategies are used to reach a constant video quality between different videos from different sources. The importance of using scalable video coding through layers scheduling is considered for video quality issues between heterogeneous users.

1.2. Contributions

The media streaming architectures studied in this work are inspired by the Mutualcast approach [49], which is an efficient mechanism for one-to-many content distribution that maximizes the overall throughput by exploiting the upload capacity of all participating peers. In this work the Mutualcast approach is extended to a hierarchical and multi-source approach. This thesis also proposes a multicast approach based on multiple m-ary trees as an alternative solution to the Mutualcast approach.

More specifically, this thesis makes the following contributions:

- Implementation of a Mutualcast prototype on the PlanetLab infrastructure which allows to evaluate the Mutualcast performance in terms of overall throughput and endto-end delay using several PlanetLab nodes located in different sites around the world.
- A hierarchical collaborative multicast model which achieves shorter end-to-end delivery time, improved scalability at reduced resource consumption. The proposed scheme involves cooperation among the participating peers which are organized into small mesh clusters at each level of a single distribution tree. This model combines the benefits from tree and mesh structures in order to reach a best possible performance in terms of delay and scalability. From tree-based distribution, the hierarchical multicast model has taken the idea of how to construct a single global tree structure to achieve

scalability, while from mesh-based approaches, the fully-connected topology to build small clusters at each level of the tree has been taken, taking advantage of the higher transmission capacities among neighboring peers. Each peer inside a cluster is a receiving and forwarding peer at the same time. Due to the fact that the upload capacity of all peers is also used, the bandwidth consumption from the source can be reduced. An analytical study and an experimental prototype of the hierarchical collaborative multicast model are presented. The experimental prototype is implemented over the PlanetLab infrastructure in order to evaluate the performance of the proposed model in real-world scenarios. The results from PlanetLab show that the proposed hierarchical multicast approach achieves a good performance, short delivery time and scalability.

- A multi-source video multicast framework for video streaming over P2P networks, which distributes multiple videos to all requesting peers exploiting full collaboration among the participating peers. In this approach, each source distributes its own video sequence while additionally forwarding video data received from other sources. A single peer is selected to redistribute a particular video block to the peers which would like to receive the videos. The goal is to maximize the overall throughput or alternatively the aggregate video quality of multiple concurrent streaming sessions. The special cases of "same throughput" or "same video quality" for all streams are also considered. The rate allocation and redistribution are formulated as an optimization problem and the proposed framework is evaluated for three different scenarios. In the first scenario, the rate allocation is jointly decided for all participating peers. In the second, the rate allocation is also decided jointly, but additionally either same rate or same video quality streams are enforced. In this scenario an adaptive mechanism is introduced in each source in order to enforce the same video quality between different video sequences. The third scenario assumes separate distribution for every source which divides its upload capacity equally among the different video sequences. The rate allocation is formulated as an optimization problem, and an analytical model is developed. To validate the analytical framework with real world environments, an experimental prototype is implemented and evaluated on the PlanetLab infrastructure. The experimental results validate the analytical analysis and show the superior performance of joint rate allocation compared to independent allocation and the effectiveness of the proposed framework.
- A multicast scheme based on multiple m-ary trees rooted at the source as an alternative to the Mutualcast approach. The proposed scheme maximizes the overall throughput while minimizing the end-to-end delay by exploiting the full upload capacities of the participating peers and their proximity relationship. The delivery scheme is based on cooperation between the source, the content-requesting peers and the helper peers. The source splits the content into several blocks and feeds them into multiple m-ary trees rooted at the source. Every peer contributes its upload capacity by being a forwarding peer in at least one of the m-ary trees. An algorithm to build the collection of m-ary trees is proposed. The proposed model is studied both analytically and by means of extensive simulation. A prototype of the proposed model has been implemented and evaluated over the PlanetLab infrastructure. The results obtained from the experiments on PlanetLab validate the results obtained from simulation and show that the proposed model achieves similar throughput as the best known solution in the literature (Mutualcast) while at the same time reducing content delivery delay.

The proposed schemes in this dissertation use a fixed network topology. However, different bandwidth allocation strategies are proposed in order to effectively deal with dynamic changes in the network condition, such as variations in the upload bandwidth, packet loss and transmission jitter. These strategies are supported by the TCP protocol. All links between nodes are established using TCP connections. Desired characteristic, such as reliable data delivery, flow-control and handling of node leave events are automatically been taken care of by the TCP protocol. The proposed schemes are directly related to small-scale Application Layer Multicast (ALM) approaches for scenarios where the number of participants is typically small such as multi-party video conferencing or multiplayer gaming.

1.3. Organization

This thesis is organized as follows. Chapter 2 provides a brief overview of peer-to-peer networks, some basics of video streaming, content delivery and scalable video. In Chapter 3, recent advances in the field of peer-to-peer streaming and multicast schemes are described. This chapter also provides an overview of Mutualcast [49], which is the reference multicast model used in this thesis. Chapter 4 focuses on the impact of end-to-end delay and the scalability during the media delivery. A hierarchical multicast model which combines a tree structure with small fully connected clusters is proposed to deal with these problems. The performance of hierarchical multicast and Mutualcast are analyzed over a real-world network. In Chapter 5, the Mutualcast approach is extended to a multi-source model to distribute different video sequences from multiple senders to multiple receivers using a fully connected network. An adaptive mechanism based on scalable video coding is introduced to reach a similar video quality between different video sequences. A description about the operations of this particular mechanism is given. This chapter also gives a detailed throughput and rate-distortion analysis for the proposed multi-source multicast model, and evaluates its performance over a real-world wide-area network using the PlanetLab testbed. Chapter 6 introduces a multicast model based on multiple m-ary trees rooted at the source as an alternative solution to the Mutualcast approach. An algorithm to build the tree collection is presented. This algorithm combines the proximity information between the requesting peers with their upload capacity in order to obtain an optimal tree collection. Different tree collections can reach the same optimal solution, where the overall throughput is maximized while the end-to-end delay is reduced. An experimental prototype is implemented on PlanetLab in order to evaluate the performance of the proposed approach. Chapter 7, concludes this thesis with a summary and some directions for future research.

This dissertation also includes an appendix. Appendix A contains additional information about the different simulations for multi-source video multicast and the results obtained from the JSVM (Joint Scalable Video Model) software which is used in this thesis to encode the test sequences for the experiments.

1.4. Performance Metrics

As previously stated, video streaming has demanding throughput, delay and loss rate requirements. For P2P video multicast, it is difficult to efficiently support video multicast while satisfying quality requirements from the users. To become highly accepted by the users, the P2P streaming systems should offer a high video quality, as well as low end-toend delay. Thus, the metrics of interest for the study of the P2P video multicast systems in this thesis are: the maximum throughput reached by the system, which refers to the overall throughput, the time to actually download the video which refers to the delivery time or end-to-end delay, and the peak signal-to-noise ratio (PSNR), which refers to the video quality.

Overall throughput. One of the key performance metrics for a distribution system is the throughput, which describe its capacity to support a given multimedia application. In [1] the throughput is defined as the number of binary digits that the network is capable of accepting and delivering per unit of time. The throughput also is sometimes called bit rate, transfer rate or data rate. In the proposed architectures, the overall throughput is related to the capacity of the network to achieve a certain distribution rate.

Delivery time. Delay is an important factor in the distribution of real-time video, because video data must be played out continuously. Due to its nature, a peer-to-peer system often needs to relay the packets along multi-hop paths to reach the end-users. This fact introduces an additional delay in each hop, which increases the end-to-end delay particularly when the links are congested. Delay variation also is a critical parameter, because the video frames must be presented to the users at a constant rate [1], and any variation must be corrected before displaying the video to the viewers.

Video quality. Video quality is a strong factor for user satisfaction. In this thesis, the peak signal-to-noise ratio (PSNR) is used to evaluate the video quality. PSNR is the most widely used objective video quality metric. A video sequence with a PSNR value between 30 and 40 dB usually is acceptable [75], while a video with a PSNR below 30 dB is typically quite bad.

1. Introduction

2. Background in Media Delivery

Multimedia refers to information that uses a combination of different content forms such as text, images, audio and video. Multimedia has experienced tremendous growth in the late 20th century and early 21st century, and it has become one of the most used terms in different scientific, economic and technological fields. In the computing field, "digital multimedia concerns with the computer-controlled integration of text, audio, still images, animation, video, and interactive content forms, where each type of information is represented, stored, transmitted, and processed digitally" [1]. This chapter describes the terms that are associated with media delivery and present an overview about the different types of communication infrastructures that are used to distribute multimedia content.

This chapter is organized in the following way:

• Media representation and compression

The key concepts of media representation and the compression standards used in this dissertation are introduced. The most popular applications in the current multimedia wave are identified.

• Media delivery

The different communication modes and protocols for media delivery are reviewed. The different alternatives for media distribution such as IP Multicast, Content Distribution Networks and P2P networks are introduced. The benefits, challenges and applications of P2P networks are discussed.

2.1. Media Representation

The presentation of the information in a multimedia system is controlled by computers, which handle the information as a succession of binary digits. To this end, all type of information such as text, images, audio and video must be coded in bits to be represented in a digital format. For the case of text, "a unit is a block of characters, where each character is represented by a fixed number of bits known as codeword " [76]. Similarly, a digital image is represented by a two-dimensional block, where each element is represented by a fixed number of bits known as codeword " [76]. Similarly, a digital image is represented by a two-dimensional block, where each element is represented by a fixed number of bits. In contrast, analog signals of audio must be converted to digital format using digitization techniques known as signal encoding. Similar to audio, a digital video can be obtained from a analog signal by using signal encoder. However, currently a video also can be obtained digital directly using a digital camera. All digital cameras use an image sensor called CCD (charge-coupled device) with photosensitive surface, which "comprises a 2-D array of sensors, each corresponding to one pixel, where the optical signal that reaches each sensor is converted to an electronic signal" [75].

Generally, the produced digital signal from these applications has a relatively high rate, which is measured in bit per seconds (bps). Due to these very high rates, these applications (uncompressed audio and video) consume a tremendous amount of storage and band-

width, which cannot be supported by the communication systems and compression techniques must be used.

2.2. Media Compression

Compression is an important issue to be considered for digital media distribution, because it helps to reduce the high resulting bit rate from the digital conversion to a level which can be supported by the communication networks. Compression removes the inherent redundancies in digitized audio and video signals in order to reduce the amount of data that needs to be stored and transmitted. In the following, this study only concentrates on video compression. To this end, a considerable effort has been realized during the last three decades for the development of video compression techniques. Thus, different standards for video compression such as H.261 [166], MPEG-1 [168], H.262 (MPEG-2) [169], H.263 [167] and MPEG-4 [170], have been developed. A recent video compression standard is H.264/AVC [171]. The results presented in this dissertation are based on the video coding standard H.264 and one of its extensions known as Scalable Video Coding (SVC) project [177]. A brief overview about the H.264/MPEG4-AVC and H.264/SVC standards will be given in the next sections.

2.2.1. H.264/AVC Video Coding

H.264/Advanced Video Coding (commonly referred to as H.264/AVC) is the newest video coding standard in the series of international video coding standards developed by the ITU-T Video Coding Experts Group and the ISO/IEC Moving Picture Experts Group [171]. The standardization of H.264/AVC, also called MPEG-4/Advanced Video Coding was completed in May of 2003 [173].

H.264/AVC offers a wide range of bit rates, frame rates and spatial resolutions, making it very versatile and guaranteeing its functionality for diverse applications [172], [175] such as broadcast over various types of networks, interactive or serial storage, video on demand or multimedia streaming. To provide flexibility and customizability for heterogeneous applications and networks, H.264/AVC integrates a Video Coding Layer (VCL), which represents the video data efficiently. The Network Abstraction Layer (NAL) formats the VCL to make it compatible with different transport layers and storage devices and provides the necessary header information in an appropriate format [171].

In comparison to the MPEG-2 standard, H.264/AVC achieves on average a bit rate reduction of up to a 50% for the same level of perceived video quality [172]. This gain results by combining a number of new features introduced by H.264/AVC, which allow it to compress video much more effectively than previous standards. Such key features include a better motion-compensated prediction with multiple reference frames for prediction, variable block-size motion compensation with finer granularity, spatial intra coding prediction capacity, 4x4 integer transform, content-adaptive in-loop deblocking filter and improved entropy coding [172].

A detailed description of H.264/AVC can be found in [171], [173], [174], [172], [175]. Additionally, a reference software implementation of H.264 is freely available in [105].

2.2.2. Scalable Video Coding

The capability of recovering important image or video information by decoding only parts of a compressed bit stream is known as scalability [75]. Using scalable video coding (SVC), parts of a video stream can be removed and the resulting substream forms another valid video stream for some target decoder. This substream represents the source content with a reconstruction quality that is less than quality of the complete original video stream but is high when considering the lower quantity of remaining data [178]. Currently, the heterogeneous networks is catching on and appears to be the trend in which networks will converge in the future. Under this conditions, SVC emerges as a valuable solution to easily adapt the coding speed to the different transmission rates and physical devices.

SVC represents the video into one base layer (BL) and one or more enhancement layers (EL), where each EL cumulatively upgrades the video quality contributed by the BL. In essence, SVC encodes the video once, into a compressed bit-stream. Each of the different users can now extract just the right amount of data from this common video stream, depending on the network conditions or device type. The main constraint is that to decode a particular enhancement layer, the BL and all the lower enhancement layers are required.

The usual forms of scalability are quality, spatial and temporal scalability. Quality scalability is more commonly known as SNR scalability. In SNR scalability, the video is represented by different layers varying in the level of perceptual quality. Here, decoding the base layer provides a low-quality of the reconstructed video. However, the resulting quality of the reconstructed video is increased by decoding the enhancement layers. In spatial scalability, the BL and the EL usually use different spatial image resolutions (resolution of the EL higher than that of the BL), making it possible for devices which have limits on their screen size (eg. PDAs), to be able to receive the video, by receiving the layer with a lower resolution. Finally, temporal scalability enables different temporal resolutions, or frame rates to represent the same video. This solution allows users with slower data links to access the same video content, but with slower speeds.

In general, each form of scalability provides multiple video representations in different resolutions, and each video representation has different importance and bandwidth requirements. The base layer (BL) is more important than the enhanced layers. Due to the coarser quality of the base layer, the base layer needs less transmission bandwidth. In contrast, the enhanced layers require more transmission bandwidth due to their finer quality. Scalable Video Coding has been used by various video streaming systems [218], [157], [146] to provide video quality under variable network conditions.

2.2.3. H.264/SVC Scalable Video Coding Extension

H.264/AVC is now a well established video coding standard and several derivative standardization projects are emerging from it. Motivated by the significant improvements introduced by the H.264/AVC standard, the Joint Video Team (JVT) of the ITU-T VCEG and the ISO/IEC MPEG have standardized a Scalable Video Coding (SVC) extension of the H.264/AVC standard [176], [177]. H.264/SVC retains most of the existing features of the H.264/AVC, and adds new components to support the temporal, spatial and quality scalability. In fact, the base layer of the H.264/SVC bit-stream is compatible with the H.264/AVC standard and can be decoded by a regular AVC decoder. H.264/SVC has achieved significant improvements in coding efficiency with an increased degree of supported scalability relative to the scalable profiles of prior video coding standards.

H.264/SVC introduces new tools which reduce the loss in coding efficiency compared to single-layer coding. Schwarz et al. state the following features as the most important innovations of H.264/SVC [178]:

- "A hierarchical prediction structure which provides temporal scalability with several layers while improving the coding efficiency and increasing the effectiveness of quality and spatial scalable coding".
- "Mechanisms for inter-layer prediction of macroblock modes, motion, and residual in order to improve the coding efficiency for spatial scalable and quality scalable coding".
- "Introduction of the key pictures concept, in oder to adjust a suitable tradeoff between drift [179] and enhancement layer coding efficiency for hierarchical prediction structures".
- "A single motion compensation loop provides a decoder complexity close to single-layer coding for spatial and SNR scalability".
- "A modified decoding process which allows a lossless and low-complexity rewriting of a SNR scalable bit stream into a bit stream that conforms to a non-scalable H.264/AVC profile".

Based on these innovations, H.264/SVC reaches a competitive rate-distortion performance while only requiring a single motion compensation loop at the decoder side. Temporal scalability is achieved using a hierarchical prediction structure [178]. The data corresponding to non-reference frames or pictures are removed from the bitstream. Hence the base layer, the one at the lowest temporal resolution is composed of only the key pictures or the key frames. For supporting spatial scalable coding, SVC follows the conventional approach of multilayer coding used by the previous standards. SNR scalability is enabled by performing residual quantization. The base layer of the SNR scalable stream is encoded using transform coding as for the H.264/AVC standard. This makes the obtained base layer compatible with the AVC standard. For the enhancement layers, the quantization error between the base layer and the original video is re-quantized, with greater precision by decreasing the quantization step size [181]. This is done by lowering the quantization parameter (QP) of the encoder. The newly constructed enhancement layer in combination with the base layer form the new base layer for the next higher enhancement layer, and afterwards the process is repeated. The refinement information is sent using progressive refinement (PR) slices. Each PR slice represents a particular amount of refinement and can be truncated arbitrarily. The received portion of the PR slice provides the improvement in the video quality.

The reference software to implement the H.264/SVC standard is the Joint Scalable Video Model (JSVM) reference software. A version of the JSVM software, written in C++ is freely available, and it can be accessed from a CVS server set up at the Rheinisch-Westfälische Technische Hochschule (RWTH) Aachen [182].

More detailed information about H.264/SVC can be found in [176], [178], [179], [180], [177].

2.3. Digital Video Formats

For many years now, digitization of video has been used in television studios, and several conversion processes between different video formats have been defined. In order to standardize these processes, the International Telecommunication Union - Radiocommunications Branch (ITU-R) developed the Recommendation CCIR-601 (BT.601) [183], which defines a standard for the digitization of video pictures. The original digitization format used in Recommendation CCIR-601 is the 4:2:2 format. Based on the 4:2:2 format, a number of variant formats such as 4:2:0 or the HDTV format have been derived [76]. In addition to CCIR-601, other digital video formats have been defined for use in other application fields such as video telephony, videoconferencing, and high definition TV (HDTV). The Common Intermediate Format, known as CIF, was introduced by the International Telecommunications Union - Telecommunication sector (ITU-T) for videoconferencing applications over ISDN/Internet. This format has about half the resolution of BT.601(4:2:0 format) in both horizontal and vertical dimensions (352x288 pixels). The quarter CIF (QCIF) has been defined for videophone and similar application. QCIF has been derived from the CIF and has half the spatial resolution of CIF in both horizontal and vertical directions (176x144 pixels). In this dissertation, the CIF and QCIF formats are used in the experiments. For example, the spatial scalability uses the QCIF (176x144 pixels) format to encode the base layer, while the enhancement layers are encoded by using the CIF (352x288 pixels) format.

2.4. Media Applications

There are several applications that involve multimedia. Some of these applications can be interpersonal and interactive communications such as telephony or videoconferencing, while others can be entertainment applications such as video on demand or television. In some of these applications just a single type of media (e.g., speech, image, text or video) is involved, while in others two or more types of media can be integrated together (e.g. videophone). Interactive applications, such as videophone and videoconferencing, require that the media be captured and encoded in real-time. In many non-interactive applications media content is pre-encoded and stored for later enjoying. In these cases, the media may be stored locally or remotely. Examples of local storage include video DVD and music CD, while an example of remote storage is video streaming over the Internet. Here, a real-time encoding is not required and a more efficient encoding can be enabled (e.g., DVD content). However, pre-encoded media has a limited flexibility, because it can not be significantly adapted to channels that support different bit rates or to clients that support different display capabilities [73]. On the other hand, some non-interactive applications, such as the live broadcast of a sporting event, may also require real-time encoding. Recently, the most visible services in the new multimedia wave are:

Voice over IP (VoIP) is an application for the transmission of voice through the Internet or other packet-switched networks. Voice over IP also is known as IP telephony. Similar to a data transfer application, the telephone signals are carried over the Internet as digital signals reduced in rate by a speech compression technique. A successful VoIP application based on P2P technology is Skype [20]. Although VoIP implementations are enjoying high popularity, they face several challenges such as latency, jitter, available bandwidth, packet loss and security.

Entertainment content distribution over Internet. With the advances in digital media and Internet, people experience video and audio contents on the Web. This fact has generated a technological revolution in video distribution and various systems to improve content

distribution have been proposed. However, video distribution on the web still has several challenges such as quality, security, reach and costs.

Videoconferencing is an example where multiple type of media can be integrated together. Videoconferencing often involves a group of people at each participating site, and it can be bi-party and multi-party. For example, during a virtual seminar, speech, video and electronic documents are integrated and transmitted to a remote group. In contrast, the information from a remote group may integrate speech and video or just speech.

2.5. Media Delivery

2.5.1. Communication Modes

The typical communication modes in today's communication networks can be classified as: unicast, broadcast and multicast. Figure 2.1 illustrates these communication schemes.



Figure 2.1.: Different distribution schemes: a). Unicast, b). Broadcast, c). Multicast.

Unicast represents a common communication form between two entities. Unicast communication also is known as point-to-point or one-to-one communication. Unicast communication can be simplex, half-duplex or duplex. Telephone conversations and video streaming over the Internet [73] are typical unicast examples.

Broadcast means that the information emitted from a source will be received by all the other devices connected at the same network. Broadcast probably represents the most popular communication scheme due to its wide usage in broadcast television.

Multicast is similar to broadcast except that the information emitted from a source is only received by a specific group of nodes in the network, which is called a multicast group. Multicast is an alternative to unicast that reduces the network traffic and optimizes the server resources [72]. Multicast is a one-to-many communication scheme, while broadcast is an one-to-all communication. Videoconferencing is a multicast example, where a predefined group of devices/computers are involved to receive the same content.

2.5.2. Media Delivery Modes

Media streams can be transmitted in two different modes: download mode and streaming mode.
Media delivery based on file download

In the download mode, the users have to download the entire media file before playing it back. Video download is similar to a file download, which allows to use established delivery mechanisms such as TCP or FTP [73]. However, the media files generally are very large files which require long transfer times and large storage capacities. The download mode requires patience from the users, who have to wait until the entire video has been downloaded before it can be viewed. Download also offers reduced flexibility, because the users must download the entire video before deciding if it is the wanted video.

Media delivery based on streaming

In contrast, in the media streaming mode, the receiver can already consume the media file while part of it is being received and decoded. In others words, media streaming allows us to reduce the delay between the start of delivery and the beginning of playback at the viewer. This delay usually is in the order of 5-15 seconds [73]. On the other hand, since only a small portion of the video is stored by the viewer during media streaming, the requirements of storage are low. However, video streaming is sensitive to the delay, because the video packets must arrive at the receiver before their playout deadlines [75]. Thus, media streaming typically requires high data rate, low-latency, or high throughput in order to offer video quality to the viewers. This is challenging since today's Internet does not provide any Quality of Service (QoS) guarantees to video streaming [74]. Additionally, the current Internet has limitations to efficiently support multicast video while providing service flexibility to users with different QoS requirements.

2.5.3. Protocols for Media Streaming

In this section, an overview about the most important Internet protocols for media streaming is presented. Specifically, the study describes the transport-layer protocols known as UDP and TCP, and how the media streaming is affected in both protocols. A detailed revision of the protocol stack for media streaming can be found in [76], [75].

Current streaming services use the User Datagram Protocol (UDP) which allows predictable and reduced delay, but it does not guarantee packet delivery.

TCP presents a number of important advantages for media streaming such as rate control and guaranteed delivery. TCP rate control has proven stability and scalability while the guaranteed delivery is achieved via retransmission. On the other hand, TCP has some practical difficulties of long delay and instantaneous fluctuations [73]. The long delay introduced by the TCP retransmission is unacceptable for real-time video applications with stringent delivery time. However, it is not always the case. For example, for Internet video streaming and broadcast applications, the allowed delay can be relaxed to few seconds and several retransmissions can be allowed [75].

Recently, "TCP-friendly" that avoids congestion similar to TCP, but without the instantaneous fluctuations has been proposed in [184], [185]. TCP-friendly rate control can coexist with other TCP-based applications and has more predictable stability and scalability properties compared to previously deployed control schemes. However, TCP-friendly rate control inherits from TCP characteristics such as the dependence of round-trip time for transmission that are unacceptable for media streaming. A detailed information about the TCP-friendly congestion control is given in [186].

The video streaming architectures proposed in this dissertation are based on TCP, in order to take advantage of its suitable flow control and congestion control mechanisms.

2.6. Delivery Infrastructures

Media streaming over the Internet has gained significant popularity in recent years due to the continuous increase in network access speed of the end-users. In this section, the main media delivery infrastructures such as IP-Multicast, Content Delivery Networks, Application Layer Multicast, and P2P Networks are introduced.

2.6.1. IP-Multicast

IP Multicast was proposed by Deering [81], as an efficient solution for one-to-many content dissemination. Multicast presents a better efficiency than unicast due to its reduced transmission overhead on the sender and the network, which reduces the delivery time for content distribution. A unicast-based distribution alternative requires that the source sends an individual stream to each end-user, which is critical for high-bandwidth applications, such as video, where large portion of bandwidth for a single stream is required. IP Multicast reduces traffic by simultaneously distributing a single copy to potentially thousands of endusers, while the multicast packets are replicated in the network by routers [83]. Different applications such as videoconferencing, distance learning and news are based on multicast technology. Figure 2.2 shows how data is distributed from a source to several end-users using IP multicast. Here, a multicast distribution tree is formed by a tree whose root is the source of the multicast tree and whose branches form a spanning tree through the network to the end-users [83]. This tree is known as shortest path tree because the shortest paths through the network are used.



Figure 2.2.: IP Multicast

IP multicast has shown to be an efficient and high performance technology for data delivery from a source to a large number of receivers. Unfortunately, IP multicast has not been fully deployed in today's Internet. Some reason for this are:

- Its deployment requires router support from all ISPs (Internet service providers), which implicates substantial modifications of the Internet infrastructure.
- New issues associated with network control and management [85] such as end-to-end reliability and flow and congestion control.

Thus, the deployment of IP Multicast has been limited to local area networks, individual campuses, and a handful of ISPs only [84]. To face these challenges, different alternative IP multicast approaches have been proposed [88], [89], [90]. Although these solutions simplify IP multicast implementation and improve the network management, they still maintain the router dependency. Recently, new IP Multicast [84], [86], [87] approaches have been proposed to provide multicast service on the Internet using native IP Multicast. Most of these solutions build IP multicast "islands", which are connected with each other using unicast tunnels.

On the other hand, IP multicast presents specific challenges to streaming media systems such as heterogeneity and more restricted choice for error control. Heterogeneity mainly originates because different receivers experience different channel conditions, while restricted choice for error control is because the retransmission in IP multicast is limited due to scalability issues [73].

2.6.2. Content Delivery Network

In today's Internet, one-to-many media streaming applications are based on the traditional client-server model of Content Delivery Networks (CDN), and commercial solutions such as Akami [78], Limelight Networks, VitalStream and other are offered via a CDN system. A CDN is formed by content servers networked together across the Internet, which cooperate with each other to transparently distribute content to end-users. Typically, the content servers are located near the users, in order to be able to serve the requested content rapidly [79]. The content servers in the CDN are connected to content providers via an internal network, which is used to transfer content from providers to content servers.

However, the CDN approach faces a number of problems such as single point of failure and costly access to high rate networks. Also, even the large streaming servers are not able to feed more than a few hundred streaming sessions simultaneously. Additionally, the selection of the best streaming server in a CDN during a session is difficult. These constrains limit the CDN's performance.

2.6.3. Application Layer Multicast (ALM)

In recent years, many researchers have chosen to use application-level solutions as an alternative to implement multicast [85], [91], [92]. In Application-Layer Multicast (ALM), all multicast tasks are implemented at the end-hosts exclusively while the network infrastructure is maintained. Figure 2.3 depicts an example ALM, where the numbers indicate the link delays.

End system multicast is present in systems where all functionality is pushed to the end hosts actually participating in the multicast group [77]. An example of this type of systems are the P2P network. An overview about P2P networks will be given in the next section.



Figure 2.3.: Application Layer Multicast

A drawback in the ALM system is its performance penalty associated with dynamic and heterogeneous Internet environments [77], because the performance is affected by the locations and stability of the end-users.

2.6.4. P2P Networks

2.6.4.1. Basic Principles

Since the Napster [2] advent in 1999, a new wave of P2P network architectures such as Gnutella [3], KazaA [5], BitTorrent [8] and many others have been deployed. Currently, the P2P networks generate an important portion of the Internet traffic. According to the Cisco study forecasts [6], the P2P networks traffic represented up to 62 percent of all Internet traffic in 2006. This trend is not expected to decrease over the next years. Instead, the Cisco report considers that the P2P traffic will increase from 1,330 petabytes per month in 2006 to 5,270 petabytes per month in 2011. The Peer-to-Peer paradigm has been a very attractive topic for many researcher from different areas such as networking, distributed systems, complexity theory, databases and others. Therefore, the Peer-to-Peer concept is quite broad and a considerable number of different definitions of "Peer-to-Peer" are found in the literature [9], [7], [24], [27], [25].

A P2P communication infrastructure is formed by a group of nodes located in a physical network. These nodes build a network abstraction on top of the physical network known as an overlay network, which is independent of the underlying physical network. Figure 2.4 shows this scenario [7], [17]. The overlay network is established by each P2P system through TCP or HTTP connections. Due to the abstraction layer TCP protocol stack, the physical connections are not reflected by the overlay network. The overlay network builds logical tunnels between pairs of nodes [11], in order to implement its own routing mechanism to transport its messages.

In the traditional client-server model two types of nodes are employed: clients and servers. In this context, clients only request services and the server only provides the clients with the appropriate service. A server can accept several requests, process them, and return the requested contents to the clients. In today's Internet, the clients include web browsers, online chat clients and email clients, while the servers typically are web servers, ftp servers and mail



Figure 2.4.: P2P Overlay Architecture

servers. In contrast, in the P2P systems, no dedicated infrastructure is required. Dedicated servers and clients do not exist because each peer can take the role of both a server and of a client at the same time. An important advantage of peer-to-peer systems is that all available resources are provided by the peers. During media distribution, peers contribute their resources to relay the media to others. Thus, as a new peer arrives to the P2P system, the demand is increased, but the overall capacity too. This is not possible in a client-server model with a fixed number of servers.

2.6.4.2. P2P Classification

Currently, P2P systems are classified mainly into two categories: unstructured and structured. This classification is based on how the nodes in the overlay structure are connected to each other.

Most unstructured P2P systems are considered as the first generation of P2P systems, which were used to share and to store files. An unstructured P2P network is formed when the logical links among participating nodes are established randomly. These systems are designed more specifically for heterogeneous and distributed environments [15], where maintaining strict restrictions on control data placement and the network topology is not possible. In an unstructured network, the queries have to be flooded through the network, which causes a high amount of traffic in the network. Unstructured systems can be further divided in [17]: centralized P2P, pure P2P and hybrid P2P. Figure 2.5 shows the architectures for these different unstructured P2P approaches.

In centralized systems such as Napster [2] a central directory server exists. This central server is responsible for answering the queries. Hence, all the query traffic is directed to it. On the other hand, in a pure P2P system all peers are equal and no peer holds any permanent information about which objects are stored where. No directory with the data of the peers which are a part of the network exists. Examples of this architecture is Gnutella 0.4 [3] and FreeNet [19]. Finally, a hybrid architecture attempts to strike a balance between the accuracy of the centralized architecture and the lower load of the pure architecture. The Gnutella v0.6 [23], JXTA [21], FastTrack [18], KaZaA [5], or Skype [20] are example of hybrid architectures.



Figure 2.5.: A comparison among the different unstructured P2P architectures. *a*). Centralized P2P, *b*). *Pure P2P, c*). *Hybrid P2P*

Structured P2P systems represent the second generation of P2P systems [13]. These systems are structured because they maintain a close coupling between the network topology and the location of data via a hash table (DHT). Ratnasamy et al. [14] define a hash table as a data structure that efficiently maps 'keys' onto 'values' and serves as a core building block in " the implementation of software systems". In fact, the hash table is used to precisely define the data placement and lookup operations. In these systems, each peer acts as a server for a subset of values, which can be data items or pointers to where data is available [27]. A DHT mechanism is responsible to handle peers joining/leaving the overlay. This DHT mechanism should be fully distributed in order to avoid a single point of failure or bottlenecks. A routing table with links to a small subset of peers is maintained in each peer. Then, these links can be used to construct the overlay network, which usually follows a regular topology such as ring, tree, mesh, or hypercube [27], [26]. Chord [4], Pastry [12], Tapestry [13] and CAN [14] are some well know examples of structured systems. All these approaches are based on hash tables and they fundamentally only use a different routing mechanism. Structured systems show superior scalability and reliability compared to unstructured systems. On the other hand, a disadvantage of structured P2P systems is their high dependence on the network dynamic, which hinders to maintain a required structure for efficiently routing messages [9], [17]. A detailed comparison between structured and unstructured is discussed in [16].

2.6.4.3. Benefits of P2P Networks

P2P systems are different to traditional client-server model. The main benefits of P2P networks are:

Decentralization. In a traditional client-server model, the information is concentrated in a central server, which distributes the information to clients through a network. In contrast, a decentralized system distributes its resources among several nodes in the system. Decentralization is a key feature in P2P networks that must be considered when the systems and applications are designed. Decentralization increases autonomy.

Cost reduction. In contrast to the centralized systems, the P2P systems provide structure to spread the cost over all participating peers [31]. In a file-sharing systems such as Napster or Gnutella, the file storage cost are shared among different peers. Thus, the P2P paradigm can have a positive impact in the business process because the costs are reduced and the

productivity is increased.

Resource aggregation. Due to its decentralized nature, the P2P networks can support the aggregation of resources. Each peer in the P2P system contributes additional resources such as upload capacity, processing power or storage capacity, which is not possible in a clientserver architecture. Large applications such as computer-intensive simulations can obtain enormous benefits from resource aggregation, because additional processing power and storage capacity from other peers is available [31]. Resource aggregation also benefits multicast applications [48], [49], because each requesting peers contributes its upload capacity to redistribute the streamed content to other peers on the same network, thereby helping to reduce the load of the server. Performance is strongly associated to resource aggregation. Performance in a system based on client-server model is affected when the number of clients is increased. In contrast, performance is improved in the P2P systems by aggregating resources such as computing power, distributed storage or upload capacity from all participating peers. However, this performance may be limited when a central server coordinates all peers (e.g., Napster) or when excessive traffic is generated by using flooding (e.g., Gnutella). To optimize performance in P2P systems different solutions such as hierarchical coordination [67], replication [65], caching [66] and intelligent routing [69], [68] have been proposed.

Scalability is the ability of a system to handle an amount of work without a considerable impact on its performance. Different types of scalability such as structural scalability and load scalability are defined in [59]. Bondi defines the structural scalability as *"the ability of a system to expand its structure without modifying its architecture* and load scalability as *the ability of a system to perform graceful as the offered traffic is increased"*. Schemes based on a central server present limited scalability mainly caused by the amount of centralized operations that need to be performed [7]. Although, scalability is improved in decentralized systems, it plays an important role when the P2P systems are designed. In the P2P networks, scalability or the number of nodes within the system (structural scalability) [31]. Napster shows a better scalability than the system based on client-server model, because in Napster the music files are directly downloaded from the peers that possess the requested content. However, pure P2P systems do not scale well, because its search is based on a flooding mechanism [27], which increases the traffic as the number of users increases.

Dynamism. P2P systems have a topology with highly dynamic behavior, where the peers enter and leave the network at any time. This implies that resources such as power computing, data or store capacity are highly variable. P2P environments fit particularly those distributed applications with highly dynamic environment such as instant messaging or distributed computing [7], which cannot be supported in a centralized environment. Dynamism is related with the ad-hoc connectivity of P2P system.

Fault resilience. Due to the decentralized nature of P2P systems, no central point of failure exist, and the vulnerability of the network as a whole is very small [17]. Nevertheless, P2P networks still face failures associated with disconnections, unreachable peer, partitions and peer failures. An active collaboration among the still connected peers is desirable to face the presence of such failures [7]. To this end, different collaboration strategies such as relays node [44], replication [64], popularity [70] and grid computing solutions [71] have been proposed.

Self-organization. The term self-organization is used in different areas of science with different meaning [61], and it has no general accepted definition in the literature. Self-organization allows the internal organization of a system, without being controlled by an external entity. A self-organization system has inherent features such as emergent behavior, adaptability, robustness[62], and it shows a high level of scalability. Self-organization is required in the P2P systems to minimize the need for configuration due to scalability, highly dynamic environments, fault resilience and cost of ownership [7], [63]. Examples of systems and products that address self-organization are Oceanstore [64] and Pastry [12].

Anonymity. In a computer system, anonymity allows users to exchange resources while preserving their privacy and remaining anonymous. Pfitzmann et al. [60] classify the anonymity between each communicating pair as sender anonymity, receiver anonymity and mutual anonymity. Anonymity is difficult to ensure in a centralized system, because all clients typically have been identified before to receive a service. One goal of P2P systems is to allow anonymity, and a particular type of P2P system called anonymous P2P has emerged in recent years. In these systems, the users are anonymous by default to service providers. FreeNet [19] represents an example of how anonymity can be implemented into P2P networks. Anonymity can be enforced in a multicast system by creating a special group formed for parties who wish to preserve their privacy and remain anonymous [7].

2.6.4.4. P2P Applications

Since the Napster advent in 1999, a significant number of P2P applications have been developed. The main drivers are the technological advances achieved in the today's personal computers and the continuous increase in network access speed of the end-users. Thus, P2P applications have become a large category, which have been categorized by some authors into major areas such as file sharing, collaboration and distributed computing [9], [7], [31], [17], [33]. In the following, an overview about these categorizations and the potential of P2P networks for media streaming delivery is discussed.

File sharing. File sharing can be seen as a form of content distribution [33], [7]. In fact, content storage and exchange have been the first applications based on P2P technology. These applications were introduced by Napster, and have become one of the areas where the P2P networks have shown a great popularity. Several content sharing applications such as eMule [22], KaZaA [5] and BitTorrent [8] still represent an important traffic portion in today's Internet.

Distributed computing. Distributed computing is a term used to indicate that different parts of a program run simultaneously on different computers, which are communicated with a network. During the last years, the grid computing concept [34] has been developed as an alternative for large scale simulations and data analyses. The term grid refers to an infrastructure that allows for the global integration of computation-intensive resources such as supercomputer clusters and storage systems, which are managed by different institutions around the world. Computing grid is a concept that can be seen as distributed computing [42]. The Genome@Home [35] and BOINC (Berkeley Open Infrastructure for Network Computing) [37] projects are examples of distributed computing applications. In Genome@Home the design of the large numbers of protein sequences is supported by distributed computing [36], while BOINC developed as an extension from the Seti@Home project [38], [40], [39], is a platform for public-resource distributed computing [41]. Although grid systems inte-

grate more powerful resources and show a more stable environment than the typical P2P systems [43], P2P technology represents a potential solution for the grid applications. This assumption is based on the fact that P2P technology takes advantage of the processing power available on the peers. Thus, a large computational job can be divided into many small independent parts, which are distributed across multiple computers. The combined processing power of peers helps to obtain faster completion time in comparison to the traditional systems. After this, the small results are integrated into the final solution.

Collaboration. Collaborating computing is an application category where individuals or teams in different geographic locations interact with one another through computation tools to share, modify, or jointly produce data and information. The collaborative applications range from online games to shared applications that can be used in business, educational, and design environments. Generally, these applications are based on events [7]. Thus, when a change occurs at one node (e.g., a modification introduced by a person), an event is generated and sent to all participating nodes. Groove [44], BSCW [45](Basic Support Cooperative Work) and Ocules CO [46],[47] are examples of collaborative applications over the Internet. Examples of collaboration for content distribution are proposed in [48],[49], [93]. In these cases, the requesting peers cooperate with each other to distribute content. Due to its adhoc nature, the P2P paradigm represents a valuable technology for user-level collaborative applications. However, these applications have stringent requirements about security and integrity [31], [7], which are formidable challenges.

Media streaming. In recent years, media streaming over the Internet has become possible, in large part due to the development of new computing technology, compression standards, high-capacity storage devices and the increase in network access speed of the end-users [74]. This fact has generated an increasing demand for multimedia services on the web, which has received tremendous attention from academia and industry. Thus, different video streaming applications for live streaming or video on demand services have emerged as valuable tools to improve communication. Streaming video applications have had a strong impact in different scenarios such as videoconferencing distribution, news distribution or event broadcast. P2P paradigm has become a promising solution for video streaming, because it offers characteristics which cannot be provided by the client-server model. P2P networks do not have a single point of failure, the upload capacity is shared among all peers, the bottlenecks are avoided, the contents can be shared by all participating peers, and they provide scalability. Subsequently, many P2P media streaming systems such as ZigZag [28], CoolStreaming [51] or Mutualcast [49], have been developed. A detailed revision of these systems is given in Chapter 3. Recently, a new media streaming application called P2PTV has been proposed as an alternative to IPTV. P2PTV combines P2P technology with Internet TV, which constitutes an important progress of large-scale P2P streaming. Many different P2PTV applications such as SopCast [52], PPLive [53], PPStream [54] have appeared on the Internet. Although P2P networks are at the moment one of the most valuable infrastructures to distribute media streaming, the P2P technology still has many open issues that must be solved to achieve high and constant video quality. To this end, new P2P media streaming applications are being investigated and developed. This thesis explores one of the media streaming applications known as P2P Video Multicast.

2.7. Chapter Summary

Multimedia content delivery over the Internet has gained popularity during the past years. However, the expansion of these multimedia services has been limited due to lack of network infrastructure or unacceptable video quality. This fact motivated the need for new compression standards and networks infrastructures. In this chapter different key concepts about media compression and delivery infrastructures have been discussed. Specifically, the most recently standards for video compression are discusses. Thus, a brief summary of video compression standards H.264/AVC and H264/SVC is presented. H.264/SVC is an emerging standard that introduces new tools which reduce the loss in coding compared to single-layer coding. Furthermore, different video formats such as CIF and QCIF has been derived from the original recommendation CCIR-601 developed by ITU-R. CIF and QCIF formats are used by the architectures presented in this dissertation.

Video can be delivered to the users via video streaming mode or video download. Video delivery based on streaming presents the best performance, with respect to video delivery based on download, because when using video streaming the receiver can already consume the media file while part of it is being received and decoded. In addition, video streaming helps to save bandwidth and storage capacity.

Delivery infrastructures play an important role for the efficient content distribution over the Internet. IP Multicast has been proposed as an efficient solution for one-to-many content dissemination. However, deployment of IP Multicast is not widely available in today's Internet. Due to this problem, many researchers have chosen to use application-level solutions as an alternative to implement multicast. In Application-Layer Multicast (ALM), all multicast tasks are implemented at the end-hosts exclusively while the network infrastructure is maintained. Recently, P2P networks have emerged as a valuable infrastructure for media streaming. This chapter concludes with an introduction to the P2P networks. The main applications and benefits of P2P networks have been discussed. File sharing, distributed computing, collaboration and content distribution are some applications of the P2P networks. P2P networks are a valuable architecture for media streaming distribution, but media streaming over P2P networks still have many open issues such as scalability, delay and video quality that must be solved before to achieve a total acceptation.

3. State of the Art in Overlay-based Distribution

In recent years many people from academia and industry have been interested in how to efficiently distribute video from one sender to many viewers, and different systems such as CoolStreaming [51], CoopNet [48] and others have been proposed and developed. Although IP Multicast has been proposed as an efficient solution for one-to-many content dissemination, several reasons such as routers migration, complexity of the protocol and a poor interoperability with existing services [197] have limited its widespread deployment in today's Internet. In such circumstances, many researchers have chosen to use application-level multicast (ALM) solutions [91] as an attractive alternative to IP Multicast for content distribution. Application-level multicast is ideal in systems where all functionality is pushed to the end hosts such as in the P2P networks. The main goal of a P2P multicast system is to maximize delivered overall quality and the overall throughput associated with content distribution to a group of viewers. This chapter explores the benefits of the overlay P2P networks for multicast services and study their categorizations from the point of view of its network topology. A brief summary of the existing P2P streaming applications according to this classification is presented. The chapter ends with an overview of existing works about multi-source multicast.

This chapter is organized as follows:

Introduction

The main challenges to build overlay P2P networks are discussed. Based on the overlay network structure, the overlay P2P systems can be classified as tree-based, forest-based and meshed-based [150], [151], [221]. An overview of these categories is provided.

Tree-based overlay for media streaming

A tree-based overlay is built using a single distribution tree rooted at the source. In this section, a media streaming model based on a single multicast tree: ZigZag [28] is discussed.

• Forest-based overlay for media streaming

The forest-based overlay takes the idea from end-system multicast [92] and expands it by organizing the requesting peers into multiple trees [97], [217]. This section gives a brief revision of SplitStream [110] and Dagster [51] which are media streaming models based on multiple trees.

• Mesh-based overlay for media streaming

In a mesh-based P2P overlay approach, the participating peers form a connected mesh and employ a delivery strategy based on swarming [97]. An example of a mesh-based overlay is Mutualcast [49], which is an efficient mechanism for content distribution in the P2P networks. A detailed revision of Mutualcast is given.

• Multi-Source Multicast

A promising approach to maximizing delivery quality to a group of requesting peers is to allow the peers to receive content from multiple sources [108], [93]. This section

focuses on the existing works on various aspects of multi-source multicast.

The contributions in this chapter are:

- 1. A revision of the different P2P overlay structures used for media streaming delivery.
- 2. Implementation and test of a Mutualcast prototype on the PlanetLab infrastructure to evaluate the Mutualcast performance under real world conditions.

3.1. Introduction

A P2P media delivery system involves two important components: an overlay network and a content delivery mechanism [108]. An overlay network is constructed over the underlying physical IP network [58] using a mechanism that determines how the peers are connected. The content delivery mechanism is responsible for streaming the content to each peer through the overlay network.

An overlay P2P multicast mainly presents the following advantages over IP multicast:

- 1. No router support is required.
- 2. More flexibility and adaptability to diverse requirements from the applications [27].

Zheng et al. [27] state that to construct and maintain an efficient overlay P2P network, mainly three problems should be considered. The first question is related to the P2P network architecture, and defines what topology is used to build the overlay network. The second question concerns network management and how the participating peers in the multicast group are managed, especially when the users present heterogeneous capacities and behaviors. The third problem is related to adaptability of the overlay network to route and schedule media data in an Internet environment whose links have an unpredictable behavior. Several solutions have been proposed to address these issues.

Three main types of overlay P2P topologies are considered for providing P2P multicast [108], [58], [101]: tree-based topology, forest-based topology and mesh topology. These topologies are illustrated in Figure 3.1. In the following, a brief survey of these overlay P2P classifications is given.



Figure 3.1.: A comparison between: a) tree-based overlay, b) forest-based overlay, and c) mesh-based overlay

3.1.1. Tree-based P2P Streaming

In the tree-based approach, an overlay construction mechanism organizes participating peers into a single tree whose root is located at the source node. The participating peers are organized as interior nodes or leaf nodes into a single tree. In Figure 3.1a, the source S sends the data to requesting peer R_1 , which forwards the data to requesting peers R_2 and R_3 . Then, a video stream in this configuration is basically pushed from a parent router to their children routers along a well-defined route. Thus, in Figure 3.1a the upload capacity of peer R_1 is used by the multicast tree for content distribution, while the upload capacity of the leaf peers R_2 and R_3 is not used. Although a tree approach probably represents the most effective distribution structure in term of bandwidth and delay optimization [109], this configuration has an inherent drawback because all the burden generated by forwarding multicast messages is carried out by a relative small number of interior nodes.

A tree structure for data distribution works well when the interior nodes are dedicated infrastructure routers as is the case for IP multicast. However, tree configuration has an additional problem in application-level multicast, where all functionality is pushed to the end-users and many of these participating peers may not have the network capacity and availability required of an interior node in high-bandwidth multicast applications [110]. This situation is shown in Figure 2.3, where end-host 2 contributes its upload capacity to forward the received content to end-host 4, while the number of copies emitted by the source is reduced. Compared with naive unicast, an efficient distribution scheme can be obtained without changes to routers, because all routing tasks are pushed to the end-users. However, the availability and capacity of end-host 2 must be considered before in order to make an intelligent construction of overlay trees and obtain a better performance of the multicast system.

3.1.2. Forest-based Overlay

In a cooperative environment such as P2P networks, the participating peers contribute resources proportional to the benefits they obtain from the system. Specifically, in applicationlayer multicast, the peers expect that the forwarding load will be shared among all participants [110]. However, multicast based on a single tree does not match well with these cooperation expectations, because the forwarding multicast traffic is carried by a small number of interior peers, while the upload capacities of a large number of leaf peers is not used. This is a critical problem for applications with high-bandwidth requirements such as video or bulk file distribution, because many interior nodes in the multicast tree may not have the required upload capacity. To face these challenges, a forest-based overlay architecture for media streaming has been proposed. A forest-based overlay organizes participating peers into multiple trees [58], and distributes the forwarding load among them in an efficient manner. In a forest-based overlay, each peer determines a proper number of trees to join based on its upload capacity. Figure 3.1b shows an example of a forest-based overlay. Here, participating peers are organized into multiple trees. To this end, each peer is placed as an internal node in at least one tree and as a leaf node in other trees. In Figure 3.1b, source S stripes its content and distribute the stripes using two separate multicast trees T_1 and T_2 . Each internal node in each distribution tree forwards any received stripe to all of their child nodes. Thus, the tree construction algorithm represents an important component of the forest-based P2P streaming approach. A strategy based on multiple trees minimizes the effect of churn [97] and optimizes the usage of available resources in the system. However, the determination of the number of required trees to maximize the overall throughput is an open problem [126].

An alternative to maximizing the overall throughput is to generate and combine all trees for all co-existing sessions in the overlay network in order to determine the optimal multicast tree collection. To this end, the source splits the content into blocks and assigns only a block to each peer for its redelivery. Every peer generates all possible trees to deliver its block to the rest of the requesting peers. If all the blocks are distributed in a parallel way several sessions coexist at the same time and all these trees generated in each node need to be combined. Thus, given *n* nodes, first all multicast trees are generated in every node using the Cayley's formula [132] and encoded using Prüfer's coding [127], [133]. Second, all multicast trees generated from all the participating nodes are combined. Third, the maximum throughput for every tree collection can be obtained using linear programming [106]. The results show that for 4 nodes (with upload capacity in kbps of 100, 200, 300 and 400) there are 6766 optimal tree collections with an overall throughput of 333.33 kbps. Only the case of 3 and 4 nodes is simulated, because the number of combinations grows exponentially. For n = 3, 4, 5, 6, the number of combination is 27, 65536, 30517578125 and 4.738E+18, respectively. Thus, finding an optimal collection of multicast trees is considered a NP-hard problem [49]. Additionally, to determine on the fly which is the optimal tree used by each node to redistribute its block within 6766 combinations is a hard task, because some nodes can use a binary tree, while others use a chain. A detailed version of these experiments is given in [128].

3.1.3. Mesh-based Overlay

Although in a forest-based overlay, a peer can join different multicast trees and receive data from different sources, they still continue to receive a given strip from a single sender, resulting in the same problem like the case of a single tree [92], [58]. These limitations have motivated a new approach known as mesh overlay networks, which is an approach inspired by BitTorrent [8], [118] or Bullet [112]. Figure 3.1b, presents an example mesh-based overlay. A mesh-based overlay is formed by participating peers via random connections, where each peer (except the source) tries to maintain a certain number of parent peers and also serves a specific number of child peers using a swarming mechanism for content delivery [95]. In a mesh-based overlay, a peer can concurrently receive data from different senders, each contributing a portion of its upload capacity. Additionally, the requesting peers can send and also receive data from each other.

The data delivery strategy for a mesh-based overlay and for a forest-based overlay is very different, because video data in an mesh-based overlay is available in multiple neighboring peers, and a node has to pull data to avoid significant redundancies while the data in a forest-based overlay is pushed from a parent peer to many child peers. Chu et al. [92], states that a good mesh has the following two properties: First, *"the quality of the path between any pair of members is comparable to the quality of the unicast path between that pair of members"*. Second, *"each member has a limited number of neighbors in the mesh"*. Here, delay and bandwidth are referred to as the application metrics, while a limited number of neighbors in the mesh helps to maintain a low overhead in the mesh.

Due to the dynamic and unpredictable behavior of peers, the main challenge of a meshbased overlay is how to select the proper senders [58] and how to cooperate and schedule the received data in the requesting peers. Recently, considerable research effort about meshbased P2P video streaming systems has been made. In [116], the authors exploit a strategy based on buffer maps to monitor network-wide quality. Carra et al. [111] present a methodology based on stochastic graph theory for mesh overlay streaming systems, and show that mesh-based overlays are very robust to failures and provide bounds on the receiving delay. Magharei et al. in [97] realize a detailed comparative study between mesh-based and multiple-tree-based approaches for live P2P streaming.

3.2. Application of Tree-based Overlay for Media Streaming

During the last years, several application-level multicast systems, e.g., Overcast [10], CAN [14], NICE [91], Bayeux [113] based on a single multicast tree have been proposed. A representative media streaming system based on a single tree is Zig-Zag [28]. In the following, a brief introduction to this system is given.

3.2.1. ZigZag

In ZigZag, the receivers are organized into a hierarchy of clusters and the multicast tree is built over this hierarchy using a set of connectivity rules called C-rules [28]. Each cluster has two heads, the first head is responsible for monitoring the memberships of the clusters, while the second head or associated-head is responsible for transmitting the content to cluster members. This organization allows for a continuous service during a failure, because in case the associate-head departs, the head can assign a new associate-head quickly. Initially, when the number of peers is small, they are organized into one cluster only. The organization will grow or shrink as new peers join or leave. ZigZag builds the multicast tree based on this administrative organization and following the three C-rules [28]. The first rule states that "a peer, when at its highest layer, neither has a link out nor a link in", while the second rule defines that "non-head members of a cluster must receive the content directly from its associatehead" and the third rule states that "the associate-head of a cluster, except for the server, must get the content directly from a foreign". The procedure to map peers into the administrative organization and the C-rules are the main contributions of this system. In [28] the ZigZag system is simulated for a set of 5000 clients under no failure and failure scenarios. Each cluster has between 5 and 15 peers, and all peers forward the content to no more than 13 other peers. The authors compare the performance between ZigZag and NICE [91], which is a P2P streaming solution based on an hierarchical arrangement of clusters. The results show that ZigZag has a maximum degree only half of the maximum degree of NICE, which means that ZigZag exhibits a less bottleneck than NICE. ZigZag and NICE are compared using other metrics such as control overhead, join overhead, failure overhead and link stress. The results show that ZigZag presents the best performance for most of the metrics.

3.3. Application of Forest-based Overlays for Media Streaming

Several application-level multicast systems based on the forest approach can be found in the literature. CopNet [48], SplitStream [110] and Dagster are examples of forest-based P2P streaming systems. The section presents a brief overview of SplitStream and Dagster.

3.3.1. SplitStream

SplitStream is an example for an efficient cooperative distribution scheme, because it distributes the forwarding load among the participating peers. To this end, SplitStream introduces the idea of multiple trees. SplitStream splits the content and distributes the striped data using separate muticast trees. Here, a decentralized scheme is used to construct a forest of multicast trees from a single source. Peers join as many trees as the number of stripes they wish to receive, and they specify an upper bound on the number of stripes they are willing to forward [110]. Thus, the key challenge in SplitStream is to efficiently construct a forest of multicast trees such that the forwarding load can be spread across all participating peers, subject to the upload capacity constraints of the participating nodes in a decentralized, scalable, and self-organizing manner. In order to balance the forwarding load among all the nodes, SplitStream accommodates nodes with different network capacities and asymmetric bandwidth on the inbound and outbound network paths, such that an interior node in one tree is a leaf node in all the remaining trees. Thus, ideally, any given peer is an interior node in only one tree and its departure can cause the temporary loss of at most one of the stripes. Multiple trees combined with appropriate data encoding such as multiple description coding (MDC) help to mitigate the effects of node failure and sudden node departure, allowing an improved robustness of SplitStream. SplitStream bases its design on Pastry [12] and Scribe [115], which are a structured P2P overlay network and a scalable applicationlevel multicast system, respectively. From Scribe, SplitStream adopts the idea of using a separate multicast tree for each of the k stripes, while from Pastry, it has taken the properties of routing to construct trees with disjoint sets of interior nodes. However, SplitStream fails to utilize the full upload capacity of all the participating peers in the multicast group, limiting the maximum overall throughput.

3.3.2. Dagster

Dagster [55] has been proposed as an end-host multicast scheme for delivering non-interactive media, which addresses issues of heterogeneity, transience of peers and cooperation. Dagster has three distinct features [55]:

- 1. The requesting peers are organized into an overlay based on a directed cyclic graph (DAG) [121]. Thus a peer can receive content from two or more parents, improving the system robustness.
- 2. Since peers receive media data from multiple peers with different bit-rates, the peers transcode the received media data before forwarding it to other peers.
- 3. An incentive system is used in order to encourage peers to share their upload capacity. The incentives are based on "(*a*) lower service rejection probability" and (*b*) "lower disruption probability".

To construct the overlay DAG, the author defines four rules: monotonically decreasing bit rate, equal bit-rate from parents, preemption, and continuous service. Based on these rules, the overlay DAG construction is realized in five main steps. To enable media streaming from multiple parents to a single peer, Dagster uses a coding method known as multiple state encoding. Dagster has a dynamic reconfiguration, because it allows that peers periodically probe the source to search for a better parent so that donations from newer peers can be fully utilized [55]. The performance of Dagster is compared with Best Fit algorithm [56]. Best Fit

inserts a newly arrival peer into a parent with largest available bandwidth, and rejects a peer if there does not exist a parent with enough bandwidth to satisfy the required bandwidth. The rejection rate is used as metric to compare both strategies. The results show that Dagster achieves a rejection rate less than 10 percent of the clients, while the Best Fit rejection rate is around 50 percent. Others metrics evaluated by Dagster are the diameter of the graph, donation level, and optimal number of parents. Diameter of the graph refers to hops away from the source. A lower diameter is desirable for media streaming in order to reduce the probability of service disruption and fluctuation. Dagster is promising in this sense, because it presents a lower diameter of graph and reduced average level of nodes. Dagster shows that there is a clear relationship between the level of a peer and its donated upload capacity when the peer is contributor-aware. An important finding in Dagster is the reduced number of parents required. The author suggests that three parents for each node is enough to reach an acceptable frame rate if one of the parents has failed. Although Dagster is a promising media streaming architecture, its topology still needs to be improved in terms of link stress and stretch.

3.3.3. M-ary Trees

This section introduces a special type of tree known as m-ary tree, in which every node has m or fewer children [133]. An m-ary tree is also sometimes known as a k-tree, N-ary tree, or a d-ary tree. A rooted tree is an m-ary tree if each internal node has no more than m children. Figure 3.2 shows an example of an m-ary tree with m = 3 for 12 nodes.



Figure 3.2.: Example of an m-ary tree for 12 nodes

Thus, an m-ary tree *T* is a finite set of nodes with the following properties [129], [130]:

- 1. it is either empty T = 0 with degree 0; or
- 2. consists of a distinguished node (the root *R*) together with an ordered *m*-tuple of subtrees, each of which is an m-ary tree, such that $T = (R, T_0, ..., T_{N-1})$ and degree N.

The empty trees are called external nodes or leaves, while the non-empty trees are called internal nodes [130].

A roted tree is a complete m-ary tree if each internal node has exactly m-children. Figure 3.2 shows an example of a complete m-ary tree for m = 3.

Additional properties of m-ary trees are [135], [130], [131]:

• For m = 2, the m-ary tree corresponds to a binary tree.



Figure 3.3.: Example of a complete m-ary tree for m = 3

- An m-ary tree is full if each internal node has *m* children.
- An m-ary tree is balanced if all external nodes or leaves *l* fall on the same level or two adjacent levels.
- An m-ary tree is complete if all external nodes or leaves *l* lie at the same level.
- The maximum number of external nodes or leaves in an m-ary tree of height h is at most m^h
- A complete m-ary tree with *i* internal nodes contains N = m(i) + 1 nodes.
- A complete m-ary tree with N nodes has i = (N 1)/m internal nodes and $l = \frac{(m-1)(N+1)}{m}$ external nodes or leaves.

Several applications of m-ary trees are found in the scientific literature. Gessel and Seo [134] present a refinement of Cayley's formula for trees using m-ary trees. In computer science, m-ary trees are used to analyze the structure of computer programs [130] in order to improve external sorting [136] or searching [137], [138]. In [123], a collection of modified m-ary trees is used as a strategy to reduce the end-to-end delay during content distribution.

3.4. Application of Mesh-based Overlays for Media Streaming

Inspired by BitTorrent [8], several mesh-based overlays for massive parallel content distribution among peers such as CoolStreaming [51], PPlive [53], Bullet [112], VMesh [117] and Pulse [114] have been proposed. A framework for small-scale application-layer multicast inspired by mesh-based overlays is Mutualcast [49].

3.4.1. Mutualcast

Mutualcast is as an efficient mechanism for one-to-many content distribution. Compared with previous multicast approaches, Mutualcast achieves the maximum overall throughput by fully exploiting the upload capacity of the peers.

Description

Mutualcast uses a fixed network topology and allows that peers with different capabilities distribute a different amount of content. Li et al. state three distinct features for Mutualcast [49]:

- 1. "Mutualcast splits the to-be-distributed content (e.g. a file or media stream), into many small blocks, and distributes each block separately".
- 2. "Each block of content is assigned to a single peer for redelivery."
- 3. "Mutualcast employs an optimal bandwidth allocation strategy, which is implemented via redistribution queues between the source and the requesting peers."

In Mutualcast, the content redistribution is realized by the requesting peers, helper peers or even the source itself. Mutualcast employs an optimal bandwidth allocation strategy in order to effectively deal with dynamic changes in the network condition, such as variations in the upload bandwidth, packet loss and transmission jitter.

The basic distribution framework of Mutualcast is shown in Figure 3.4, for one source S, three requesting peers R_1 , R_2 , R_3 and one helper peer H_1 .



Figure 3.4.: Mutualcast content distribution network

In this scheme, the peers R_1 , R_2 , R_3 request a copy of the content from the source S, while helper peer H_1 only contributes its upload capacity to help distributing the content to the other peers. When a block is assigned to requesting peers R_1 , R_2 and R_3 for redistribution (e.g., blocks 1, 2, 3 and 4), the block is first sent by the source to the peer in charge, which then forwards the block to the other two requesting peers. The helper peers are not interested in receiving the videos and just contribute their resources during distribution. When the block is assigned to helper peer H_1 for redistribution (e.g., blocks 5, 6 and 7), the block is first sent by the source to the helper peer H_1 , which forwards the block to the other three peers. When the source has abundant upload capacity, it may also choose to directly distribute the block (e.g., block 8) to each requesting peer R_1 , R_2 , R_3 . Thus, Mutualcast uses three distribution routes to distribute content [49]:

- 1. Through requesting peers,
- 2. Through helper peers, and
- 3. Directly from the source.

Mutualcast chops the content into a number of small blocks for distribution and assigns each block to a certain node for redistribution. The number of blocks assigned to a peer is proportional to its upload capacity. Mutualcast assumes that the upload capacity of each peer is the only constraint, which is an assumption motivated by the fact that peers usually have larger download capacity than upload capacity (e.g. DSL lines) on the Internet. Mutualcast sets the size of these blocks as 1KB, in order be able to send each block as a single packet over the network.

In Mutualcast, the connections between peers are established using TCP. Mutualcast leverages several benefits from TCP such as flow control, reliable data delivery and node leave events. Additionally, the TCP buffers are the redistribution queues, which solves the problem of establishing buffer queues to send content among peers and to control the speed of distribution between them.

Analytical Framework

[49] presents an analytical framework for Mutualcast which will be briefly reproduced in the following and later extended. The analysis assumes that the Mutualcast scheme is formed by a source S of upload capacity C_S , N_1 requesting peers denoted as R_i with upload capacity C_{R_i} , and N_2 helper peers H_i with upload capacity C_{H_i} . The participating peers for the Mutualcast scheme are shown in Figure 3.5. For Mutualcast, the exhaustion of the source



S: Source with upload capacity C_S

 N_1 : Content requesting peers $R_1 \dots R_{N_1}$ with upload capacity C_{R_i}

 $N_2\!\!:\! \text{Helper peers } H_1\!\cdots\!H_{N_2}$ with upload capacity C_{H_i}

Figure 3.5.: Participating peers in the Mutualcast scheme

upload capacity is given by:

$$B_R + B_H + N_1 B_D = \sum_{i=1}^{N_1} B_{R_i} + \sum_{i=1}^{N_2} B_{H_i} + N_1 B_D$$
(3.1)

The first term represents the amount of data that is sent from the source to the N_1 content requesting peers for redistribution. The second term corresponds to the data being sent from the source to the N_2 helper peers and the last term represents the block of video data B_D that is directly sent to the N_1 content requesting peers. B_{R_i} is limited by the upload capacity C_{R_i}

$$B_{R_i} = \frac{C_{R_i}}{N_1 - 1} \tag{3.2}$$

and represents the content that has to be redistributed to the N - 1 other content requesting peers. Similarly, for the helper peers, B_{H_i} is limited by the upload capacity C_{H_i}

$$B_{H_i} = \frac{C_{H_i}}{N_1}.\tag{3.3}$$

then B_R and B_H are obtained as

$$B_R = \sum_{i=1}^{N_1} B_{R_i} = \sum_{i=1}^{N_1} \frac{C_{R_i}}{N_1 - 1} = \frac{N_1}{N_1 - 1} \frac{1}{N_1} \sum_{i=1}^{N_1} C_{R_i} = \frac{N_1}{N_1 - 1} \overline{C_R}$$
(3.4)

$$B_H = \sum_{i=1}^{N_2} B_{H_i} = \sum_{i=1}^{N_2} \frac{C_{H_i}}{N_1} = \frac{N_2}{N_1} \frac{1}{N_2} \sum_{i=1}^{N_2} C_{H_i} = \frac{N_2}{N_1} \overline{C_H}$$
(3.5)

Where $\overline{C_R}$ and $\overline{C_H}$ are the mean upload capacities of the N_1 requesting peers and the N_2 helper peers, respectively. The Mutualcast distribution throughput Θ , which represents the amount of content sent to the requesting peers per second is defined as

$$\Theta = \begin{cases} C_S & \text{for } C_S \le (B_R + B_H) \\ B_R + B_H + \frac{(C_S - B_R - B_H)}{N_1} & \text{for } C_S > (B_R + B_H) \end{cases}$$
(3.6)

When a special case with no helper nodes ($N_2 = 0$) is considered, (3.6) reduces to:

$$\Theta = \begin{cases} C_S & \text{for } C_S \le B_R \\ B_R + \frac{(C_S - B_R)}{N_1} & \text{for } C_S > B_R \end{cases}$$
(3.7)

where the overall throughput Θ for $C_S > B_R$ can also be written as

$$\Theta = \frac{N_1}{(N_1 - 1)}\overline{C_R} + \frac{C_S - \frac{N_1}{(N_1 - 1)}\overline{C_R}}{N_1} = \frac{N_1}{(N_1 - 1)}\overline{C_R} - \frac{\overline{C_R}}{(N_1 - 1)} + \frac{C_S}{N_1} = \overline{C_R} + \frac{C_S}{N_1}$$
(3.8)

Simulation

Similar to [49], Mutualcast is evaluated in terms of overall throughput and linear programming is used to maximize the overall throughput. Linear programming [107], [204] has been successfully used as an optimization tool by other authors [106], [119], [120]. A set of five peers is used, where peer *S* acts as the source and the four peers R_1 , R_2 , R_3 and R_4 work as requesting peers. The upload capacity of the participating peers C_S , C_{R_1} , C_{R_2} , C_{R_3} and C_{R_4} in kbps is 1000, 750, 1000, 750, and 500, respectively. Helper peers are not considered in this example. Figure 3.6 shows this mutualcast setup and its translation into a linear program for throughput maximization.

In Figure 3.6, the source splits the content into five blocks. The blocks X_1 to X_4 are distributed from the source to the requesting peers through other peers, while the block X_5 is distributed from the source to each requesting peer directly. This example assumes that the source has an upload capacity of 1000 kbps. In the linear program (Figure 3.6 b), the first constraint $X_1 + X_2 + X_3 + X_4 + 4X_5 \le 1000$ kbps considers the upload capacity of the source, which has to deliver one block to each requesting peer R_1 , R_2 , R_3 and R_4 , and additionally sends the block X_5 to every peer directly. The other constraints consider the upload capacity of the requesting peers R_1 , R_2 , R_3 and R_4 , which redistribute the received blocks X_1 , X_2 , X_3 and X_4 , respectively to the rest of peers in the mutualcast group. The solution gives a maximum throughput of 1000 kbps, while the size of the blocks in kbits is $X_1 = 250$, $X_2 = 333.33$,



Figure 3.6.: Mutualcast example and its translation into a linear program for throughput maximization

 $X_3 = 250$, $X_4 = 166.66$ and $X_5 = 0$, respectively. All constraints ensure that the limit of the capacity of the source and the requesting peers are not exceeded.

Four different examples using different upload capacities for the source and the requesting peers are evaluated. The results are summarized in Table 3.1.

C_S	C_{R_1}	C_{R_2}	C_{R_3}	C_{R_4}	Throughput
(kbps)	(kbps)	(kbps)	(kbps)	(kbps)	(kbps)
500	250	1000	500	1000	500
1000	750	1000	500	1000	1000
500	500	200	500	250	487.5
1000	500	750	500	750	875

Table 3.1.: The overall throughput Θ of Mutualcast

In the first two examples, $B_R > C_S$, then θ is given by the first part of (3.7), and θ become same as C_S . In contrast, in the last two examples, $B_R < C_S$, then θ is given by the second part of (3.7).

Limitations

However Mutualcast incurs performance penalties because, depending on the locations of peers, packets are likely to travel along sub-optimal paths, and packets may traverse the same links multiple times. Thus, proximity and delay are important issues to be considered. Since all nodes in Mutualcast are fully connected, the scalability also is limited.

Extensions

Mutualcast has been extended in several works [140], [139]. In [139], Mutualcast is used for a multi-party real-time audio conferencing system. To this end, the peers form a fully connected group and during the conference, every peer takes turn to mix and redeliver the compressed audio. The audio is split into frames, which are mixed and redelivered by the peers. Each peer can redeliver a number of frames proportional to its upload capacity. Huang et al. [140] propose an adaptive rate control scheme for streaming media in Mutualcast. This approach uses multiple bit rates to gracefully adjust the common quality received at all peers.

3.5. Hybrid Tree/Mesh Overlay

Recently, new hybrid overlay ALM structures have been proposed. These hybrid tree/mesh overlays combine the advantages of tree-based and mesh-based approaches for media delivery. Examples of hybrid overlays are MeshTree [122], mTreebone [109] and hierarchical collaborative multicast [50]. Because the full knowledge of the network topology in a distributed environment is not available, a solution based on greedy often leads to overlay ALM structures that are inefficiently structured. MeshTree address the greedy problem. To this end, MeshTree introduces two key ideas [122]. First, the overlay structure must contain a low cost tree which connects nodes that are topologically close together. This tree rooted at the source is called the backbone tree. Second, shortcut links must be added on top of the tree in order to improve the delay of the backbone tree. This creates a mesh overlay, which is degree-bounded based on the individual peers' capacity. Similar to MeshTree, mTreebone uses a backbone tree, but it is constructed over a set of static peers. These static peers organized with others form a mesh overlay, which facilitates to accommodate dynamic peers and fully explore the available capacity between peers. Here, the main challenge is to identify the static peers. To this end, the authors use an age threshold-based method to identify the static peers and design the backbone tree. Hierarchical collaborative multicast uses a single tree and forms small clusters in each peer. Chapter 4 gives a detailed review of this scheme.

3.6. Multi-Source Multicast

Multi-source P2P multicast applications recently have been used for collaborative environments such as conferencing, multi-player games or content distribution services [161], [163], [162]. The classical tree solution for multicast schemes shows limitations such as failure node fragility or additional delay when it is used in a multi-source context [112]. An additional limitation for tree solutions is the limited collaboration among all peers. On the other hand, the emerging P2P overlays known as unstructured and structured overlay show limitations for multi-source multicast such as scalability [141], large overhead [143] or complex protocols [142]. In [11] a new technique called Unstructured Multi-source Multicast is presented. This solution builds and maintains multicast distribution trees from many sources on top of an unstructured base overlay. The model deals with some limitations in the unstructured P2P overlay such as scalability and large overhead. The authors in [94] introduce a distributed video streaming framework, which shows the benefits of video streaming from multiple servers to a single receiver. In [144], a P2P media streaming model is proposed that involves multiple sending peers in one streaming session, which uses a tomography-based sender selection protocol to optimize the quality at the receivers. Recently, the authors in [103] and [145] have proposed new approaches for content distribution from multiple sources to a single receiver. While in [103] the authors present and evaluate an algorithm for the optimal bit allocation in combination with scalable video techniques for distributed streaming environments, in [145] the authors exploit the similar source concept to significantly improve the download time of a file from multiple sources to one receiver. All these solutions are only partially collaborative because the collaboration among the sources is not considered. In [146], PALS (P2P Adaptive Layered Streaming) is presented as a receiver-driven coordination mechanism for quality adaptive streaming from multiple congestion controlled sender peers to a single receiver.

3.7. A Global Testbed: PlanetLab

An effective way to design, evaluate, and deploy geographical distributed network services is to use a global testbed [154]. However, conventional testbeds do not support a dual use from researchers that want to develop new applications and clients that want to use these applications. To deal with these issues, PlanetLab has been proposed as a promising overlay-based testbed. PlanetLab is designed under the following principles [154]:

- "Services should be able to run continuously and access a slice of the overlay's resources",
- "Control over resources should be distributed",
- "Overlay management services should be unbundled and run in their own slices", and
- "APIs should be designed to promote application development".

How PlanetLab realizes the distributed virtualization and unbundled management is described in [155]. Since July 2002 hundreds of users have used PlanetLab for a wide range of services such as content distribution networks [158], [159], routing and multicast overlay [92], [93], [50], P2P-based video streaming [157], [51]. Currently, PlanetLab [153] consists of 843 machines, hosted at 422 sites, spanning over more than 30 countries. In [156] the authors report their experience building PlanetLab over the last years. In September 2006, a project called OneLab [160] has been started in Europe. OneLab manages the European part of the PlanetLab platform, and operates in close cooperation with the PlanetLab team.

3.7.1. Mutualcast Implementation and Evaluation

A Mutualcast prototype has been implemented and evaluated on PlanetLab. The implementation includes a sender module run by the source peer and a receiver module run by each of the requesting peers. Mutualcast implementation is based on the TCP protocol, which manages the sending and receiving buffers and the redistribution queues in all participating peers. To evaluate the performance of Mutualcast on PlanetLab, a small multicast group formed by four PlanetLab nodes is selected. The source is located at University of Pittsburg (planetlab2.cs.pitt.edu), while the requesting peers are located at University College London-UCL (planetlab1.net.research.org.uk), Worcester Polytechnic Institute-WPI (75-130-96-13.static.oxfr.ma.charter.com) and Universidad Técnica de Lisboa-UTL (planetlab-2.tagus.ist.utl.pt). The source broadcasts a media file of 11 MB to all the requesting peers. The experiments demonstrate the capability and flexibility of the Mutualcast approach. The throughput of the source and the three requesting peers are shown in Figure 3.7 a)-d). The horizontal axes represent the time during the first 30 seconds. The throughput of the source represents its sending rate, while the throughput of the requesting peers represents their reception rate. The requesting peers receive packets based on their redistribution capacity, which is limited by their upload capacity. In the experiment, the throughput of the source and the requesting peers are changing every second in response to the network changes. This adaptation allows that Mutualcast assigns more blocks to peers with better network conditions at the time.



Figure 3.7.: Mutualcast throughput measurements: a) throughput of the source, b)- d) throughput of the requesting peers.

3.8. Research Challenges

Although significant advances in P2P media streaming have been made in recent years, there are still many open issues. Yiu et al. [164] identify the problem of locating supplying peers and maintaining content delivery paths as the two major challenges in providing P2P media streaming. These challenges are presented due to the dynamic nature of the peers. In a P2P system, a supplying peer might suddenly leave or crash, and the requesting peers need to find new supplying peers. Additionally, peers have limited bandwidth capacities, and often multiple peers are also required in distributed streaming systems [94], [165]. Another important challenge in P2P media streaming is the heterogeneity of the underlying IP

networks [58]. This heterogeneity is mainly introduced by the heterogeneous receivers and asymmetric access capacities. Download and upload capacities of the peers are asymmetric, because the peers usually have larger download capacity than upload capacity (e.g. DSL lines). Download capacity decides the receiving capacity for each individual peers, while the upload capacity decides the total available capacity for the system. These characteristics may generate random packet-losses in the network or at the buffers of the peers when the access bandwidth of a peer or the sum of all supplying peers are less than the average bit rate required by the video stream [58]. Video streaming has certain requirements on bandwidth, packet loss, and packet delay. However, under this situation a P2P system does not provide any guaranteed support to video streaming services [125], and new solutions to satisfy stringent quality requirements must be investigated. A promising solution to this problem is to provide self-adaptive QoS for each individual peer according to the current network conditions, while the total upload capacity of all peers is fully utilized [58]. To this end, some issues such as content aware media data organization, priority-based media data delivery, QoS adaptive multi-source and layered media data scheduling should be addressed. On the other hand, Mutualcast [49] is proposed as an efficient multicast mechanism for content distribution from one source to multiple requesting peers. However, Mutualcast concentrates on improving the overall throughput, and does not consider the Rate-Distortion performance of video delivery. The Mutualcast concept can be extended to multiples sources in order to obtain an optimal solution that maximizes the overall throughput or alternatively the overall video quality as will be shown later in this thesis.

3.9. Chapter Summary

This chapter has introduced the benefits of overlay P2P networks for multicast services and their different categorizations. From the point of view of their network topology the overlay P2P networks have been categorized as: Tree-based overlay, forest-based overlay and meshbased overlay. The main advantages and disadvantages for each approach were discussed. Based on these concepts, several P2P media streaming systems have been developed. Some existing P2P streaming applications according to this classification such as ZigZag, Split-Stream and Dagster have been discussed in the chapter. The m-ary tree concept is introduced as an alternative to create distribution schemes. Specially, the chapter has presented a detailed revision about Mutualcast which is the reference model in this dissertation. The Mutualcast revision presents an analytical framework for this model and its simulation based on linear programming. The study also includes the Mutualcast evaluation in the Planet-Lab testbed. PlanetLab is a overlay-based testbed, which has been proposed to evaluate and deploy geographical distributed network services.

Recently, new hybrid overlay ALM structures have been proposed. These hybrid tree/mesh overlays combine the advantages of tree-based and mesh-based approaches for media delivery. Some examples of these hybrid structures were also discussed in the chapter. Due to the growing number of Internet users, some applications require multi-source multicast support. Multi-source multicast is present in collaborative environments such as videoconferencing, multi-player games or content distribution services. The chapter gives an overview of existing works about multi-source multicast. The chapter ends with a short discussion about the main research challenges in the P2P media streaming field.

4. Hierarchical Collaborative Multicast

4.1. Introduction

Mutualcast maximizes the overall throughput by exploiting the upload capacity of all participating peers. A drawback of using a fully-connected network is that the number of connections is proportional to n^2 because each peer has to forward its received blocks from the source to all other peers. Mesh-based approaches also have high control overhead due to data scheduling and limitations for delay sensitive applications when the participating peers are located in different geographical locations. This chapter presents hierarchical collaborative multicast [50], an ALM scheme that is inspired by Mutualcast and tree-based approaches. On the one hand, Mutualcast has been shown to provably maximize the overall throughput during a multicast session, while on the other hand tree architectures introduce lower end-to-end delay, are more scalable and are easier to maintain. A hierarchical structure has the advantage of high scalability. The main motivation is to design an ALM scheme that can achieve shorter end-to-end delivery time, improved scalability and low resource consumption. Thus, hierarchical collaborative multicast adopts a tree structure as the global structure but incorporates small mesh clusters on each level of the single distribution tree. Clusters are an elementary unit of a hierarchical architecture, which involves one source peer and several requesting peers. The peers inside a cluster are fully connected. Each peer inside a cluster is a receiving and forwarding peer at the same time. Due to the fact that the upload capacity of all peers is also used, the bandwidth consumption from the source can be reduced. The performance of the proposed approach is compared with Mutualcast [49] using PlanetLab. The results show that the proposed scheme achieves a better performance in terms of transfer time than Mutualcast as well as a reduced memory usage in each forwarding peer.

This chapter is organized in the following way:

• Description

In this section, the Mutualcast scheme is extended to a hierarchical concept. The design of a novel hierarchical collaborative multicast approach is introduced and its differences with respect to other multicast approches are explained.

• Clusters Organization

The requesting peers are organized into clusters, which are allocated into a distribution tree. Flow control and caching mechanisms are proposed to distribute the content between a cluster head and its cluster child. The Mutualcast redistribution mechanism must be adapted a content redistribution based on hierarchical clustering.

• Simulation

The distribution problem is formulated as an optimization problem and the model is translated to a linear programming problem for throughput maximization. The performance of the hierarchical collaborative multicast approach is simulated in terms of throughput.

• Implementation

This section explains the implementation of an experimental prototype of the hierarchical collaborative multicast approach. The performance of a hierarchical collaborative multicast prototype is evaluated in the local network of the Institute of Communication Networks (LKN) at the Technische Universität München (TUM). To verify the prototype performance in a wide-area network, the PlanetLab infrastructure is used. The performance of the proposed approach is compared with the performance of Mutualcast in terms of throughput and delay.

The contributions in this chapter are:

- 1. A hierarchical collaborative multicast approach, where the global distribution scheme organizes the requesting peers into small meshed clusters hierarchically located on each level of a single distribution tree. The approach aims at improving the scalability and delay of Mutualcast.
- 2. The formulation of rate allocation as an optimization problem with objective functions that maximize the throughput of the system by exhausting the upload capacity of all participating peers including the source. The optimization problem is translated into a linear program which is solved using optimization techniques.
- 3. An experimental prototype of the hierarchical collaborative multicast scheme, and its evaluation and comparison with the Mutualcast approach. Both approaches are evaluated in a local network and using the PlanetLab infrastructure.

4.2. Description

Mutualcast [49] is a scheme designed for small-scale overlay multicast applications that achieves the maximum possible overall throughput during a multicast session by exploiting the upload capacity of all participating peers. Although Mutualcast maximizes the throughput, it provides limited scalability and does not control the delivery time. The main motivation behind hierarchical collaborative multicast is to improve scalability issues in Mutualcast. On the other hand, the multicast-tree based content distribution has been shown to have high scalability and low delay, but the upload capacity of all participating peers is typically not fully exhausted. For example, NICE [91] adopts hierarchical clustering to achieve reduced transmission delay in a large-scale network, however, NICE fails to utilize the full upload capacity of all the participating nodes in the multicast group, limiting the maximum overall throughput. Hierarchical Collaborative Multicast has some similarity to NICE and Mutualcast. More specifically, it combines the best concepts from both approaches. From NICE, hierarchical collaborative multicast has taken the idea of how to construct the distribution tree in order to gain scalability, while from Mutualcast it has taken the fully-connected topology to maximize the overall throughput and the efficient adaptability to bandwidth variations of individual peers. The proposed approach differs from Mutualcast in that it uses clusters allocated in a unique distribution tree rooted at the source node. The hierarchical collaborative multicast structure is shown in Figure 4.1.

Each node in a cluster can potentially become the source node of new cluster. Therefore, the structure inside one cluster is fixed while the global structure is flexible. In each cluster a



Figure 4.1.: Hierarchical Collaborative Multicast architecture for 15 peers organized in clusters of size 3

fully connected topology as in Mutualcast is built, but considering proximity information. In this work the Round-Trip Time (RTT) between two peers is used as proximity information to build the local clusters. In Figure 4.1, the peers 1, 2 and 3 form the cluster with the highest hierarchy level (cluster 1) in the multicast session. In this cluster, each peer forwards its blocks received from the source to the rest of the peers within the same cluster, while receiving the rest of the blocks from them. Peers in the same cluster have a bidirectional communication among them. Parallelly, each peer in the first cluster acts as a source for a new cluster located on the second level of the hierarchical structure. Thus, peer 1 is a forwarding peer of cluster 1 and a source peer of cluster 2 (which is formed by peers 7, 8 and 9) at the same time. Peers 2 and 3 can also extend their own clusters. Cluster 2 is denoted as a child-cluster of peer 1. The communication between peers located in the first cluster and the requesting peers clustering on a second level is unidirectional. In other words, in the hierarchical approach, the blocks are distributed from one cluster to another, from top to bottom. Using local cluster, the peers can greatly benefit from the capacity of other neighboring requesting peers via local collaboration while the number of connections is reduced in comparison to a fully connected overlay topology.

In a fully connected network as Mutualcast, a logical link from each peer to every other peer is established. The number of required links *K* is computed as

$$K = N_1^2 \tag{4.1}$$

where N_1 is the number of requesting peers.

For a distribution tree with balanced clusters with the same number of peers in each one, the number of required links *K* is given by

$$K = N_1 * m \tag{4.2}$$

where m is the number of peers in the local clusters.

For example, given a multicast group with 15 requesting peers, hierarchical collaborative multicast forms a small and balanced cluster of size 3 in each level of the hierarchical structure and 45 links are only required to distribute all the blocks. In contrast to the proposed approach, using Mutualcast, the overlay network is formed with 225 connections. Thus, the hierarchical approach shows better scalability than the Mutualcast approach.

4.3. Cluster Organization

4.3.1. Cluster Formation

Clustering has received considerable attention during the past few years in the scientific community [189], [191], [193], [196], [219]. Complex networks such a social networks or the World Wide Web exhibit a high degree of clustering and scale-free [189], [192]. Ravasz and Barabási show in [190] that these two characteristics are the result of a hierarchical organization, implying that small groups of nodes organize in a hierarchical manner into large groups, while maintaining a scale-free topology. Indeed, many complex networks are modular because in these networks groups of nodes exist that are highly interconnected with each other, but have few links to nodes outside of the group to which they belong to. Perhaps one of the clearest examples of complex networks are P2P networks. The hierarchical collaborative multicast scheme assumes that clusters with high interconnected nodes combine into each other in a hierarchical network.

Hierarchical collaborative multicast first forms a hierarchical structure and then evolves into meshed clusters. In hierarchical mode, data must be distributed from one cluster to another, from top to bottom. A peer may belong to two clusters located on different layers of a hierarchical tree.

The requesting peers form a small cluster with their neighbors based on a proximity policy. Each cluster is represented by a "cluster head", while the other nodes close to the cluster head are integrated into the cluster. Figure 4.2 shows zone-based organization for the source and a cluster head.

The clustering strategy has some similarity to clustering algorithms presented in [194], [219] and [195]. In Figure 4.2 a) clusters are formed without considering proximity constraints, and peers located in distant regions can be interconnected to each other. However, high delays may be introduced in these types of clustering. In Figure 4.2 b), concentric rings are used to define different zones associated with a peer. The size of zones is based on Round Trip Time (RTT). Any peer sees the rest of the system as a set of concentric rings. Thus, the source can select some requesting peers for the top cluster from its closest rings. These peers selected by the source become the cluster head of other clusters or sub-clusters and form their associated zones. In addition, the clustering strategy associates a weight with each peer, which is defined by its upload capacity.

The clustering formation also has some similarity to k-means clustering [200], [199] because it divides the set of peers into L regions such that the proximity relative to the cluster head (centroids of the clusters) is minimized. However, the hierarchical clustering is different from k-means clustering because the number of clusters is not chosen before the procedure starts. The regions L work as landmarks for the geographical location of the peers, in order to avoid that peers distant to each other, but with the same proximity to the source, be assigned



Figure 4.2.: Zone organization during cluster formation. *a*). Proximity constraints are not considered, *b*). Concentric rings are used

to the same cluster.

The hierarchical clustering strategy works as follows:

- 1. The source *S* calculates its proximity to each requesting peer based on the round trip time (RTT).
- 2. Based on its upload capacity C_S , the source S determines a number of peers to feed directly.
- 3. Sources define a set of regions *L* to be used as landmarks.
- 4. The source selects a peer as cluster head based on the following rules:
 - cluster head is the peer with the largest upload capacity C_{R_i} in a defined region
 - cluster head has an acceptable proximity with the source
 - high connectivity and acceptable proximity between the cluster head and a potential group of requesting peers
- 5. Selected peers are fully connected to each other.
- 6. Each cluster head repeats step 1 to 6 until all requesting peers are clustered.

A source peer is the head of its cluster in each level. The number of levels and the number of peers clustered into each cluster depend on the upload capacity of the source (or head peer) and allowed delay constraints. These values are used as threshold values to determine when the procedure stops. The number of final clusters, or peers within each cluster is not known ahead of time.

To better utilize the upload capacity of the requesting peers and to obtain reduced delays, peers with higher upload capacity should be placed into clusters located in superior levels, while peers with lower upload capacity should be placed into clusters located further down the tree.

4.3.2. Flow Control Mechanism

In a multicast system, the most precious resource is the upload capacity of the source, where the content originates. This dissertation assumes that the upload capacity of each peer is the only constraint. This assumption is motived by the fact that peers usually have larger download capacity than upload capacity (e.g. DSL lines) on the Internet. Thus, the flow control mechanism is based on the upload capacity of the source and the requesting peers. Initially, the source does not know the upload bandwidth of all requesting peers. Under this scenario, the source cannot send content to all peers as quickly as possible. The reasons are the following. First, if the download capacity of a peer is much greater than its upload capacity, the packets may be overstocked, and this peer must take more and more space to store packets in the memory. Second, a peer may have a much broader upload capacity and forwards the packets immediately, and mostly it waits for the packets from the other peers. For example, the source sends packet 301 to peer R_1 and packet 302 to peer R_2 . Peer R_2 has abundant upload capacity, while R_1 has a scarce upload capacity. Peer R_2 forwards packet 302 to the other peers immediately, and is still waiting for the packet 101, which was previously received by peer R_1 . So, peer R_2 must also store packets in the range from 102 to 302 in memory and cannot play the content from packet 101 even it has received packet 302 already.

To deal with this problem, the flow control mechanism assumes that a peer does not read packets from the source link, if one of the forwarding buffers is full. This scenario is shown in Figure 4.3. Here, the forwarding buffers of node R_1 to R_2 and R_3 are full. The peer R_1 stops reading packets from source S. Then the sending buffer of the source to the peer R_1 will also be full soon, and the source will not load packets to this buffer any more. On the other hand, the peers R_2 and R_3 have large upload capacity and are reading packets from their sockets very quickly. Thus, the source can rapidly load packets to its other two sending buffers and send them to the peers R_2 and R_3 . Following the sequence number of packets in the buffers, the source sends packets much faster to peers R_2 and R_3 than to peer R_1 . If each forwarding buffer is still available for more packets, the corresponding bit in the socket related to source will read data from the socket of the connection from the source until the whole packet has been received. Then, this packet will be put into each buffer of the forward link, to be forwarded to other peers in the same cluster. The payload of this packet is stored in the memory block.

4.3.3. Caching Mechanism

In hierarchical collaborative multicast, the requesting peers need to efficiently store the payloads of the packets received from the source and other peers in the same cluster. To this end, a caching mechanism must store contents in separate storage according to the ID of the source and the requesting peers, which is labeled in the header of the packet. The design assumes that there is enough space to store the content. In each peer, the payload of the packets should be sorted by sequence number and stored continuously. However, two problems are present: media streaming size is unknown and the packets are arriving out of order from different peers.

Arrays can solve these problems. They are a standard solution used for storing data. An



Figure 4.3.: Flow control mechanism

array is a structure consisting of a group of elements that are accessed by indexing [202], [203]. This allows for very fast access because the code can do a little jump quickly to any location in the array, and the elements are all grouped together so they tend to be in memory at the same time [201]. However, the size of the array must be fixed and predefined before distribution, and is impossible to extend its size later. On the other hand, in a list, there is no fast way to access the N-th element, but the size is easy to append, which means the size does not need to be predefined before the distribution. In hierarchical collaborative multicast, the size of the content is unknown and an array can be used to store the packets according to the sequence number that is specified in the packet header. Thus it is not suitable for storing large size content. Multimedia streaming is played frame by frame, as soon as each frame is completely received, it can be read out of the buffer for playback, and the space of this frame in the memory can be released. Furthermore, the order of the incoming packets depends on the upload bandwidth of the redistributing peers, and is most probably out of order. The size of the buffer should be dynamic in order to optimize the efficiency of the memory usage.

Figure 4.4 shows how the arrays and double-linked lists are employed to store the received packets.

When using double-linked lists, a block can be inserted or removed in the middle of the list, or added at the beginning or the end. After removing or inserting that block, the linked-list should be reconnected, and the size or the index updated if necessary. In addition, the size of the list can be specified or unlimited. This mechanism minimizes the usage of memory space, and it is suitable for application of playing the stream during receiving.

In a cluster, each requesting peer receives packets from the source (via a direct link) and other members in the same cluster (via forwarding links), which have different upload capacity. The packet distance shows the difference of performance between these routes in



Figure 4.4.: Storage blocks

terms of memory usage. A large packet distance indicates that a packet must wait a long time in the memory before it can be forwarded. So, the larger the packet distance, the larger the memory usage in each peer.

The flowchart in Figure 4.5 illustrates the algorithm of storing packets in the storage blocks. Once a new packet is received, the peer checks first the sequence number that is specified in the packet header, then determines the position of the packet inside the block. Although at the source the packets are distributed in order for the requesting peers, the packets are received via different links and due to the different network capacity, they typically arrive out of order. If the specific block does not exist, the peer creates a new block, and adds it at the correct position within the double-linked list. After storing the packet the peer checks whether this block is full. If the counter is equal to the number of packets defined for this block, then the block is full and its index must be checked. If the index of the current full block is the smallest, then, this block is the first block of the list (or the previous blocks have been removed), and the data in this block can be read out. After this, this block can be removed from the list, and the peer sets the index of the next block as the smallest.

4.3.4. Redistribution Mechanism between Clusters

In hierarchical collaborative multicast, each requesting peer receives packets from the source and from the other peers in the same cluster. Each peer forwards the packets received from the source to the other requesting peers in the same cluster. If the requesting peers are cluster heads then they redistribute the received blocks to their child-clusters.

A redistribution to the children after the whole content has been received is not feasible, because the playing time difference (redistribution delay) between two levels become too



Figure 4.5.: Flow diagram for the storage process

large and the nodes need too much space to store the content. Furthermore, the last level peers experience excessive delay before they can start playing the content (potentially in the order of minutes or even hours). The required time T_{rc} to download all content to the top cluster from the source is given by

$$T_{rc} = \frac{content\ size}{\Theta} \tag{4.3}$$

where Θ represents the overall throughput of the system.

In contrast, the redistribution mechanism approach based on streaming avoid large delivery delays between clusters, and allows that the cluster heads stream the content to their child-clusters as soon as a block is ready. In this case, the delivery delay is given by

$$T_{rb} = \frac{block\ size}{\Theta} \tag{4.4}$$

again, Θ represents the overall throughput of the system.

From 4.4, it becomes obvious that playout delay becomes much shorter using block distribution, because the size of a storage block will typically be much smaller than the whole content.

Similar to Mutualcast [49], hierarchical collaborative multicast uses redistribution queues to handle anomalies such as packet loss and network congestion during content distribution. The flow chart for the forward link is shown in Figure 4.6. In this scheme, a peer removes in each iterative loop a data block from the incoming link, and copies this block onto the outgoing links connecting all the other requesting peers. The next block from the incoming link is not removed until the last data block has been successfully copied onto all delivery links. This forwarding scheme is used by a peer to redistribute the block received from the source to all the other peers in its cluster. If the peer is a cluster head, then the peer collects the blocks from all peers in the top cluster and sends these blocks to its child cluster as a source peer. Thus, in order to reduce the content receiving delay between a cluster head and its child-peers, the cluster head peer should send the content via the forwarding link after receiving some blocks instead of receiving all content from its parent peer or the root source.



Figure 4.6.: The operation flow of the forward link on requesting peer.

The delivery links of the requesting peers are the same as in Mutualcast. A block can arrive from other peers into the same cluster or from the source. Blocks to be redelivered to the other requesting peers have a higher priority than the blocks to be consumed in the local peer.
4.4. Simulation

In this section, the proposed framework is simulated using linear programming. The simulation evaluates the hierarchical collaborative multicast scheme and Mutualcast in terms of the maximum possible throughput. The set is formed by 7 participating peers, where peer S acts as the source peer and the rest of the peers R_1 to R_6 are requesting peers. The upload capacity of the source varies from 400 to 1100 kbps, while the upload capacity of the peers B_{R_1} , B_{R_2} , B_{R_3} , B_{R_4} , B_{R_5} and B_{R_6} in kbps are 1150, 1300, 150, 300, 150, and 300, respectively. Helper peers are not considered in this example. In order to maximize the overall throughput, the simulation considers that video sequences can be split in several parts or blocks. The assumptions are translated into a linear program which is shown in Figure 4.7.



Figure 4.7.: An example of the hierarchical collaborative multicast scheme for six requesting peers together with the corresponds linear program. The objective is to maximize the overall throughput

In Figure 4.7, the first constraint $X_1 + X_2 + 2X_3 \le 1000$ represents the upload capacity of source S, which has to deliver the blocks X_1 , X_2 to the requesting peers R_1 and R_2 , respectively. Source S exhausts its upload capacity by delivering a block X_3 directly to peer R_1 and R_2 . The requesting peers R_1 and R_2 form the first cluster in the first level of the distribution tree. The second constraint $2X_1 + X_2 + 2X_3 \le 750$ represents the upload capacity of requesting peer R_1 , which has to redeliver the block X_1 to peer R_2 and R_3 , block X_2 to peer R_4 , and block X_3 to peer R_3 and R_4 . Similarly, exhaustion of the upload capacity of requesting peer R_2 is represented by the third constraint. The fourth to seventh constraints represent the contribution of the requesting peers R₃, R₄, R₅ and R₆, respectively, who redistribute their received block in the local clusters. The constraints $0 \le X_1$ to $0 \le X_3$, enforce that negative block sizes are not allowed. To solve the linear program, the Mathematica 5.0 software packet was used. The solution gives a maximum throughput of 725 kbps, while the rate of the blocks in kbps is $X_1 = 150$, $X_2 = 300$ and $X_3 = 275$, respectively. Here, if the block size is zero, it means that no block is transmitted on this link. To compare the Hierarchical Collaborative Multicast scheme to Mutualcast, the participating peers are organized into a Mutualcast scheme, and the assumptions are translated into a linear program. Thus, the source sends blocks X_1 to X_6 to requesting peers R_1 to R_6 for redistribution, while block X_7 is directly sent to each requesting peer. Mutualcast leads to a maximum throughput of 725 kbps, while the rate of the blocks in kbps is $X_1 = 230$, $X_2 = 260$, $X_3 = 30$, $X_4 = 60$, $X_5 = 30$, $X_6 = 60$, $X_7 = 75$. The results show that the hierarchical collaborative multicast scheme achieves the same overall throughput as Mutualcast, while the upload capacity of the source and all requesting peers is fully exhausted. After this, the upload capacity of the source S_1 is varied from 400 to 1100 kbps, while the upload capacity of the requesting peers is maintained. The results from Mutualcast and the hierarchical collaborative multicast are summarized and compared in Figure 4.8.



Figure 4.8.: Overall throughput comparison for Mutualcast and Hierarchical Collaborative Multicast

Figure 4.8 shows that Hierarchical Collaborative Multicast reaches an overall throughput lower than Mutualcast when the upload capacity of the source S varies from 500 kbps to 900 kbps, because the upload capacity of the source cannot be used optimally. However, when the upload capacity of source S is 1000 kbps, the overall throughput reached by both schemes is identical, because the upload capacity of the source and all peers is fully exhausted. When the upload capacity of S is 400 kbps both schemes reach the same overall throughput, because the source is fully exhausted in both cases. Finally, if the upload capacity of the source S continues to grow above 1000 kbps, Hierarchical Collaborative Multicast reaches an overall throughput lower that Mutualcast. This is because in Hierarchical Collaborative Multicast the upload capacity of the cluster heads R_1 and R_2 has been exhausted and they cannot redistribute more blocks to their child-clusters. Then, source S cannot be exhausted and Hierarchical Collaborative Multicast cannot reach the same overall throughput as Mutualcast.

4.5. Implementation

The proposed hierarchical collaborative multicast scheme has been implemented using Linux and C/C++. Similar to Mutualcast, all links between nodes are established using TCP connections. Desired characteristic, such as reliable data delivery, flow-control and handling of node leave events are automatically been taken care of by the TCP protocol. In both approaches, all the peers (except the root source) send and receive packets at the same time, and the distribution of blocks among the requesting peers is implemented using threads. In computer term, a thread is a "lightweight" process, which is smaller, faster and more ma-

neuverable than a traditional process [205]. In this work, the threads are used in order to ensure distribution and storage without delay. Each node runs 3 threads in parallel: sending, receiving and measurement. According to the task of each peer in the system, they are classified as source, requesting peer or cluster head.

A "sender" module runs at the root source, which is located at the top in Figure 4.1. This module does not receive any content, and sends content only. The sender module includes two threads. The first thread called *source thread* delivers content, while the second thread, called *measurement thread* measures the upload throughput. The sender thread performs the initialization of the source and delivers the content to the requesting peers after the cluster has been established. Meanwhile, measurement thread starts measuring the upload throughput until it receives an asynchronous ending message from the sender thread. A "receiver" module runs on every requesting peer, which has no child-cluster. This request module has only one thread, which receives and forwards the content received from the source to the rest of the peers in its cluster. A cluster head peer works as a requesting peer and server peer. To this end, a "cluster head" module is run by each cluster head peer. The cluster head module combines "sender" and "receiver" modules. Thus, three threads called source thread, requesting thread and measurement thread are enabled in each cluster head. The communication between threads in provided by the redistribution queue. The requesting thread puts the storage blocks into the queue and the source thread redistributes them to the child-cluster one by one.

To run the protocol, initially, the source starts listening on a pre-defined port and waits for the socket connection request. Then, each requesting peer establishes a connection with the source. The source copies the IP address and listening port of every requesting peer and sends this information to all requesting peers. The flow diagram in Figure 4.9 shows the steps followed by the source during the system initialization.

The cluster initialization takes place in the very beginning, in order to define the hierarchical collaborative structure to be used. To this end, each requesting peer uses the IP address and Round Trip Time (RTT) to calculate its proximity to the rest of the requesting peers. The local clusters can be organized by combining proximity information with information about the upload capacity of the peers. Each peer starts a new thread, which listens and waits to establish the forward link for transferring content between two peers. All peers maintain a list of all peers in the cluster. So the forwarding connection could be established very quickly. The flowchart in Figure 4.10 shows the establishment of a connection on a requesting peer. After the initialization of the forward link, an initialization finished signal is received from the source indicating that the cluster is initialized and all links are ready for content distribution. Then, the source and the requesting peers begin the data transfer.

The source has a connection to each requesting peer, and a sending buffer is associated to each connection by the source. In these sending buffers, the source loads packets from its hard disk and sends these packets through TCP connections. After the blocks are received at each requesting peer, it forwards the block to the rest of the peers in the cluster. If the requesting peer is a cluster head, it organizes the received blocks and forwards them to its child-cluster. For this task, the requesting peers check each forwarding buffer. If forwarding buffer is still available for more packets, the source or cluster head will be informed via the socket descriptor. The source checks this information and the next packet is put into each buffer of the forward links.



Figure 4.9.: Establishment of peer connections and cluster formation from the source' perspective.

In the implementation, the block size is set to 1KB, so each block can be sent using a single TCP/IP packet. The buffer size is set to 3 KB, which corresponds to 3 packets. A loading pointer indicates how many packets remain in the buffer and how much space is available.

4.6. Evaluation

The performance of a hierarchical collaborative multicast prototype has been evaluated in the local network of the Institute of Communication Networks at the Technische Universität München and in the PlanetLab [153] infrastructure. For both infrastructures, the proposed approach and the Mutualcast approach have been compared in terms of delivery delay.

The delivery times between Mutualcast and the proposed approach are measured making extensive tests in the local network. The obtained measurements from the local network are very similar between for both schemes, due to the high capacity and homogeneity of the participating peers. In addition, the experiments compare the resource consumption (memory usage) between Mutualcast and hierarchical collaborative multicast by broadcasting a media file of 11 MB from the source to all requesting peers. The results are shown in Figure 4.11.



Figure 4.10.: Establishment of connections on the requesting peer.



Figure 4.11.: Packet distance comparison between Mutualcast and Hierarchical Collaborative Multicast

The memory usage is determined by the packet distance, which indicates the memory space required in each peer to forward the blocks received from the source to the rest of the requesting peers. The larger the packet distance, the larger the memory usage in each forwarding peer. The results show that for the same bandwidth capacity among all requesting peers, Mutualcast shows a bigger distance than Hierarchical Collaborative Multicast. Therefore, Mutualcast needs a greater usage of memory in each forwarding peer than the proposed approach.

For the experiments on PlanetLab, a simple and small multicast group formed by 10 PlanetLab nodes is selected. The tested scheme is shown in Figure 4.12. In the experiments, the source is located at University of Pittsburg (planetlab2.cs.pitt.edu), while the first level cluster in the hierarchical structure is formed by PlanetLab nodes located at University College London-UCL (planetlab1.net.research.org.uk), Worcester Polytechnic Institute-WPI (75-130-96-13.static.oxfr.ma.charter.com) and Laboratory of Computer Sciences, Paris 6-UPMC (planetlab-01.lip6.fr). Two child-clusters are built on the second level. The first child-cluster which is formed by PlanetLab nodes located in the Northeast of the United States (Massachussetts Institute of Technology (planetlab7.csail.mit.edu), University of Toronto (pl2.csl. utoronto.ca) and University of Chicago (planetlab3.cs.uchicago.edu) is connected to the WPI PlanetLab node, while the second child-cluster which is formed by PlanetLab nodes located in Poland Warsaw-UT(planetlab3.mini.pw.edu.pl), Wroclaw-UT (planetlab1.ci.pwr.wroc.pl), and TP-Warsaw (planetlab1.warsaw.rd.tp.pl) is connected to the UCL PlanetLab node. Then, the source broadcasts a media file of 1.5 MB using both schemes to all the requesting peers.



Figure 4.12.: Organization of Planetlab nodes using Hierarchical Collaborative Multicast

The first experiment on PlanetLab evaluates the adaptability of the hierarchical collabora-

tive multicast scheme to the bandwidth variation of the peers. Figure 4.13 shows how the source fits the content distribution to the peers based on their available capacity. The peers are organized into three clusters located on different levels through the distribution tree. Thus, each peer in the first cluster (WPI, UCL and LIP6) receives and forwards a part of the broadcasted file from the source based on its throughput. WPI receives and forwards 31.9 % of the total file, while UCL and UTL receive and forward 35.2 %, and 31.9 % of the total file, respectively. Similarly, in the first (cluster 2) and second (cluster 3) child-clusters, each requesting peer receives and forwards a part of the broadcasted file from its parent peer. In cluster 2, MIT receives and forwards 38.4% of the total file, while UChicago and UToronto receive and forward 33.2% and 28%, respectively. In cluster 3, TP-Warsau receives and forwards 24 % of the total file, while Wroclaw-UT 49.4% and Warsaw-UT 26.6 %, respectively. In each cluster, the number of blocks to be distributed to each peer are dynamically adapted considering its changing upload capacity.



Figure 4.13.: Dynamic load distribution in clusters

In contrast, using Mutualcast, the load distribution among the nodes is very unbalanced. Mutualcast distributes a number of blocks to each peer based on its upload capacity in order to obtain uniform content distribution. However, some peers do not contribute with enough upload capacity or the connection is very slow. The load distribution for the same peers using Mutualcast is illustrated in Figure 4.14. Here is shown how two PlanetLab nodes located in Poland (TP-Warsaw and Wroclaw-UT) receive a reduced number of data blocks (percentage of received file) compared to nodes located in the Northeast of the United States.

The second set of experiments evaluates Hierarchical Collaborative Multicast and Mutualcast in terms of delivery delay. In the experiments, the scheme with the smallest delivery delay is assumed to present the best overall performance. The maximum delay for both approaches during the delivery of 1.5 MB is shown in Figure 4.15.

In contrast to the measurements realized in the local network, the Mutualcast and Hierarchical Collaborative Multicast have different performances on PlanetLab, due to the heterogeneity of the participating peers. The results show that when the proposed approach is used, the maximum delay is smaller in comparison to Mutualcast for the most of the requesting peers. Using the proposed approach the average delay is reduced by around 40% with respect to Mutualcast.

This improvement first is attributed to the fact that the source is close to the cluster at the highest level of the hierarchy. Thus, the throughput between the source and this subset of requesting peers is larger than the throughput between the source and the rest of the requesting peers. The second fact is that using a hierarchical approach, the peers in the local



Figure 4.14.: Dynamic load distribution in Mutualcast



Figure 4.15.: Maximum delay comparison between Mutualcast and Hierarchical Collaborative Multicast

clusters avoid the connection to distant peers into the overlay topology, while when using Mutualcast all the requesting peers are fully connected. Another important contribution of hierarchical collaborative multicast is the clustering and load distribution among peers with similar performance and proximity.

4.7. Chapter Summary

In this chapter, a Hierarchical Collaborative Multicast scheme for video delivery from one source to multiple receivers using Peer-to-Peer (P2P) networks has been proposed and evaluated. Tree and mesh structures are combined for video delivery.

Hierarchical Collaborative Multicast organizes all nodes as a scalable distribution tree. Requesting peers are organized into small meshed clusters, which are hierarchically located through a unique distribution tree. Content is distributed from the root source to every peer through the tree. If a requesting peer has a child-cluster, then this peer is called a cluster-head peer. A cluster head peer and its child-peers comprise a distribution cluster. The hierarchical clustering is based on the upload capacity of the requesting peers and their mutual proximity. In each cluster, all the participating peers are fully interconnected, and they are in fact receivers and senders at the same time.

The source chops the content into many packets and sends each packet to a single peer. Then the peer exchanges the received packets among the rest of the peers to obtain the whole content. The implementation utilizes sending and forwarding buffers, a storage block and dynamic adaptive distribution mechanism in order to maximize the throughput, minimize the memory usage and redistribution delay between nodes of different levels in the distribution tree.

The local network of the Institute of Communication Networks (LKN) and the PlanetLab [153] infrastructure are used to evaluate the performance of the hierarchical collaborative multicast scheme. In both infrastructures, the performance of the proposed scheme has been compared with the Mutualcast performance. The results show the superior performance of hierarchical collaborative multicast compared to Mutualcast. Specifically, the proposed scheme presents a better load distribution, a reduced average delay and a reduced memory usage.

4. Hierarchical Collaborative Multicast

5. Multi-Source Video Multicast

In this chapter, a novel approach for video delivery from multiple sources to multiple receivers in P2P networks is introduced. The source peers, the requesting peers and additional helper peers are considered as participating peers. The approach assumes that all requesting peers and all sources need to receive all videos. The helper peers are not interested in receiving the videos and just contribute their resources during distribution. The proposed approach is inspired by Mutualcast [49], which is an efficient mechanism for one-to-many content distribution that maximizes the overall throughput by exploiting the upload capacity of all participating peers. The Mutualcast scheme is extended from one source to multiple sources and the optimal rate allocation for multi-source streaming applications is investigated.

In the scheme proposed in [93], each source distributes its own video sequence and additionally forwards blocks of video received from other sources to the rest of the requesting peers. How much the source can redistribute depends on the available upload capacity. At the same time, each requesting peer forwards the blocks directly received from a source to the rest of the peers. Again, the amount of redistributed content depends on the peers upload capacity. The optimal rate allocation among multiple streaming sessions is determined and the effectiveness of the proposed scheme is evaluated for four different scenarios. In the first scenario, the rate allocation is jointly decided for all participating peers. In the second scenario, the rate allocation is also decided jointly, but additionally either same rate or same video quality streams are enforced. In the third scenario, separate rate allocation for every source and same rate streams are assumed. In this case, the upload capacities of the sources are divided equally among the different video streams. The fourth scenario also considers sources with independent rate allocation, but for different rate streams. Again, the upload capacities of the sources are divided equally among the different video streams. The analysis assumes that the upload capacity of each peer is the only constraint, which is an assumption motivated by the fact that peers usually have larger download capacity than upload capacity (e.g. DSL lines) on the Internet. Each sources splits its video into many small blocks, and delivers each block separately. The analysis also assumes that all participating peers have heterogeneous upload capacities and may store different number of video blocks. Exhausting the upload capacity of each peer leads to the best system performance, which becomes reflected by maximum throughput or best possible video quality.

This chapter is organized in the following way:

• Description

The section starts by declaring the motivation for this work. In particular, how the Mutualcast approach can be extended to multi-source scenarios such as multi-party videoconferencing is discussed. Then, the multi-source framework is introduced and its principle design is described.

• Analytical model

An analytical framework for the multi-source video multicast scheme is presented.

The proposed scheme is analyzed in terms of throughput maximization and Rate-Distortion optimization.

• Adaptive rate allocation mechanism

Here, an adaptive rate allocation algorithm is introduced, which enforces either same rate or same video quality streams in the overall system.

• Simulation

To simulate the performance of the multi-source multicast approach, the distribution problem is formulated as an optimization problem [188] [107] and translated into a linear programming problem, which is solved using standard optimization techniques. The performance of the proposed approach is simulated in terms of throughput and video quality (PSNR).

• Implementation and evaluation

In this section, the implementation of an experimental prototype of the proposed multisource multicast approach is described. The performance of the prototype has been evaluated in a local network and in the PlanetLab infrastructure. The performance of the proposed approach has been compared to a multi-source multicast scheme with independent rate allocation in terms of throughput, PSNR, and delay for all video streams.

The author's research contributions in this chapter are:

- 1. A multi-source multicast model, targeted for video streaming environments that inherently have multiple senders. The model assumes that the senders simultaneously stream different video sequences to multiple receivers. This includes one source receiving the videos from the other sources.
- 2. An analytical framework for the proposed multi-source video multicast approach, which is divided in throughput-based optimization and Rate-Distortion-based optimization (video quality). The formulation of rate allocation as an optimization problem with objective functions that either maximize the throughput or the video quality of the system by exhausting the upload capacity of each peer. The optimization problem is translated into a linear program which is solved using common optimization techniques.
- 3. An adaptive rate allocation mechanism for the multi-source scheme that adjusts the transmission rates for both sources in order to enforce either same rate or same video quality streams in the system.
- 4. An experimental prototype of the multi-source multicast approach, and its implementation on the PlanetLab infrastructure to evaluate its performance for different scenarios: sources with joint rate allocation for arbitrary rates or arbitrary video quality streams, sources with joint rate allocation for same rate or same video quality streams, and sources with independent rate allocation.

In this work, sources that perform a joint rate allocation decision during a streaming session are referred to as sources with joint rate allocation. On the other hand, sources that work separately, for which their upload capacities are divided equally among the different video streams are referred to as sources with independent rate allocation. The results obtained from extensive simulations and experiments on PlanetLab validate the analytical study and the effectiveness of the multi-source multicast model, and show the superior performance of joint rate allocation compared to independent rate allocation.

5.1. Description

This section provides an overview of the proposed multi-source video multicast model. The solution builds on top of Mutualcast. The proposed approach differs from previous multi-source multicast approaches in that it uses a fixed network topology where all the participating peers are fully interconnected, including the sources. Also different to previous work, the solution performs a joint rate allocation decision considering the upload capacities of all participating peers. The proposed model is illustrated in Figure 5.1 for two source peers, two requesting peers and one helper peer.



Figure 5.1.: Proposed for multi-source multicast approach

In this example, the peers S_1 and S_2 are the sources, which contain the video sequences Xand Y to be distributed among all participating peers, while peers R_1 , R_2 are the requesting peers, and peer H_1 is a helper peer. The peer H_1 does not request the videos, nevertheless, it contributes its upload capacity to help distributing the videos to the other peers. Here, all the peers are in fact receivers and senders at the same time, as it is for instance the case in a multipoint video conferencing scenario. Each requesting peer forwards a received video block to the other requesting peers and the other source peers. Each source splits the original content into small blocks and one unique peer is selected to distribute a block to the rest of the peers. For this example, the source S_1 divides the video X into the blocks X_1 to X_5 , while the source S_2 divides the video Y into the blocks Y_1 to Y_5 . Because the proposed approach is based on collaboration among sources, each source distributes its own video while additionally forwarding the block of video received from the other source to the rest of the requesting peers. At the same time, each requesting peer forwards the blocks (X_1, Y_1) , and (X_2, Y_2) are assigned to the requesting peers R_1 and R_2 , respectively, while the block (X_3, Y_3) is assigned to the helper peer H_1 , the block X_4 is assigned to the source peer S_2 and block Y_4 is assigned to the source peer S_1 for distribution. Peers with different upload capacity distribute a different amount of content. The block size assigned to each requesting peer is proportional to its upload capacity. When the source peers have abundant upload resources, each source additionally sends one block directly to the video receiving peers. To illustrate such a case, Figure 5.1 shows that source S_1 directly sends block X_5 to each peer and source S_2 directly sends block Y_5 to each peer. Thus, source S_1 sends one block to each participating peer for redistribution, one block in parallel to all requesting peers, and forwards one block of the video Y received from the source S_2 to each requesting peer R_i . The source S_2 behaves similar as source S_1 , but in a complementary way. It sends the video Y and forwards the video block X_4 . Each requesting peer forwards the blocks received from the sources S_1 and S_2 to the other requesting peers and the other sources, e.g., peer R_1 receives the blocks X_1 and X_5 from source S_1 and the block Y_1 and Y_5 from the source S_2 . After this peer R_1 forwards the block X_1 and Y_1 to the rest of the participating peers except to the source where the block was originally generated and the helper peer H_1 . The blocks X_3 and Y_3 are sent by the sources S_1 and S_2 , respectively to the helper peer H_1 , which forwards the blocks to all participating peers except to the source where the block was originally generated.

5.2. Throughput-based Analysis

The analytical study of the multi-source multicast model is divided in two parts. The first part corresponds to the throughput-based analysis, which is studied in this section. The second part corresponding to the video quality analysis is studied in the next section.

The throughput-based analysis of the multi-source multicast model is realized for four scenarios:

- 1. Sources with joint rate allocation for different rate streams,
- 2. Sources with joint rate allocation for same rate streams,
- 3. Sources with independent rate allocation for same rate streams.
- 4. Sources with independent rate allocation for different rate streams.

In each scenario, a particular case and the general case are analyzed. The particular case considers two sources and N_1 requesting peers, while the general case considers M sources, N_1 requesting peers, and N_2 helper peers.

5.2.1. Sources with Joint Rate Allocation for Different Rate Streams

In the first scenario, the rate allocation is jointly decided for all participating peers. All participating peers are fully connected and all of them need to receive all videos. In order to maximize the overall throughput, video sequences with different rate are allowed.

A. Particular Case

The particular case for this scenario considers two sources and N_1 requesting peers R_i . Source S_1 and S_2 have an upload capacity C_{S_1} and C_{S_2} , respectively, and the content-requesting



Source S_1 with upload capacity C_{S_1} Source S_2 with upload capacity C_{S_2} Content requesting peers $R_1,...,R_{N_1}$ with upload capacity C_{R_i} , $i = 1, ..., N_1$

Figure 5.2.: A multi-source example with two sources and N₁ requesting peers

peers R_i have upload capacity C_{R_i} . The case for two sources and N_1 requesting peers is illustrated in Figure 5.2.

The exhaustion of the upload capacity of source S_1 is given as

$$C_{S_1} = \sum_{i=1}^{N_1} B_{R_i}^1 + B_{S_2}^1 + (N_1 + 1)B_D^1 + N_1 B_{S_1}^2$$
(5.1)

The first term in (5.1) represents the amount of data being sent from the source S_1 to the requesting peers for redistribution. $B_{S_2}^1$ is the amount of data being sent from source S_1 to source S_2 for redistribution. $(N_1 + 1)B_D^1$ stands for the upload capacity needed to directly send a video block from source S_1 to the N_1 requesting peers and the other source S_2 . $N_1B_{S_1}^2$ is the upload capacity that is needed at source S_1 to redistribute a block received from S_2 . Similarly, the exhaustion of the upload capacity of source S_2 is achieved for

$$C_{S_2} = \sum_{i=1}^{N_1} B_{R_i}^2 + B_{S_1}^2 + (N_1 + 1)B_D^2 + N_1 B_{S_2}^1$$
(5.2)

 $B_{R_i}^1 + B_{R_i}^2$ is limited by the upload capacity C_{R_i} .

$$B_{R_i}^1 + B_{R_i}^2 = \frac{C_{R_i}}{(N_1 - 1) + 1} = \frac{C_{R_i}}{N_1}$$
(5.3)

The denominator in (5.3) reflects the fact that the content has to be redistributed to the other $N_1 - 1$ requesting peers and the other source. Then, B_R becomes

$$B_R = \sum_{i=1}^{N_1} B_{R_i}^1 + \sum_{i=1}^{N_1} B_{R_i}^2 = \sum_{i=1}^{N_1} (B_{R_i}^1 + B_{R_i}^2) = \sum_{i=1}^{N_1} \frac{C_{R_i}}{N_1} = \frac{1}{N_1} \sum_{i=1}^{N_1} C_{R_i} = \overline{C_R}$$
(5.4)

If the sum of the two source upload capacities is smaller than the mean upload capacity of the requesting peers, the throughput Θ is limited by $C_{S_1} + C_{S_2}$:

$$\Theta = C_{S_1} + C_{S_2} \quad \text{for} \quad C_{S_1} + C_{S_2} < B_R = \overline{C_R} \tag{5.5}$$

If the sum of C_{S_1} and C_{S_2} is larger than $\overline{C_R}$ additional throughput can be achieved. Two additional distribution routes are distinguished:

- 1. Source S_1 redelivers $B_{S_1}^2$ from source S_2 and hence an additional upload capacity of $N_1 B_{S_1}^2$ is required. Similarly, if source S_2 redelivers $B_{S_2}^1$ from S_1 an additional upload capacity of $N_1 B_{S_2}^1$ is required at source S_2 .
- 2. Direct delivery of data to all requesting peers which requires an additional upload capacity at source S_1 of $(N_1 + 1)B_D^1$. Similarly, $(N_1 + 1)B_D^2$ is required at source S_2 .

These two alternatives put the same total burden on the sources if $B_D^1 = B_{S_2}^1$ and $B_D^2 = B_{S_2}^1$. Then, direct delivery to $N_1 + 1$ peers can be selected. Thus, if $C_{S_1} + C_{S_2} > B_R = \overline{C_R}$ the achievable throughput Θ is:

$$\Theta = B_R + \frac{(C_{S_1} + C_{S_2} - B_R)}{(N_1 + 1)}$$
(5.6)

Therefore, the distribution throughput Θ for the multi-source scheme using two sources is:

$$\Theta = \begin{cases} C_{S_1} + C_{S_2} & \text{for } C_{S_1} + C_{S_2} \le B_R \\ B_R + \frac{(C_{S_1} + C_{S_2} - B_R)}{N_1 + 1} & \text{for } C_{S_1} + C_{S_2} > B_R \end{cases}$$
(5.7)

B. General Case

For the general case M sources, N_1 requesting peers R_i and N_2 helper peers H_i are considered. Source S_i sends video data blocks to all requesting peers, all helper peers and all other sources for redistribution. Additionally, if enough upload capacity is available, a video data block B_D^i is directly sent to all other sources and the requesting peers. Exhausting the upload capacities of sources S_1, \ldots, S_M is achieved if

$$C_{S_j} = \sum_{i=1}^{N_1} B_{R_i}^j + \sum_{i=1}^{N_2} B_{H_i}^j + \sum_{i=1, i \neq j}^{M} B_{S_i}^j + (N_1 + (M-1)) B_D^j$$
(5.8)

Similar to (5.7), the distribution throughput Θ for the multi-source scheme with M sources, N_1 requesting peers R_i and N_2 helper peers H_i can be written as:

$$\Theta = \begin{cases} \sum_{i=1}^{M} C_{S_i} & \text{for } \sum_{i=1}^{M} C_{S_i} \le B_R + B_H \\ B_R + B_H + \frac{\sum_{i=1}^{M} C_{S_i} - (B_R + B_H)}{N_1 + M - 1} & \text{for } \sum_{i=1}^{M} C_{S_i} > B_R + B_H \end{cases}$$
(5.9)

with the contribution B_R from the requesting peers R_i

$$B_{R} = \sum_{i=1}^{N_{1}} B_{R_{i}}^{1} + \ldots + \sum_{i=1}^{N_{1}} B_{R_{i}}^{j} + \ldots + \sum_{i=1}^{N_{1}} B_{R_{i}}^{M} = \sum_{i=1}^{N_{1}} (B_{R_{i}}^{1} + \ldots + B_{R_{i}}^{M})$$
(5.10)
$$= \sum_{i=1}^{N_{1}} \frac{C_{R_{i}}}{N_{1} + M - 2} = \frac{N_{1}}{N_{1} + M - 2} \overline{C_{R}}$$

and, the contribution B_H from the helper peers H_i

$$B_{H} = \sum_{i=1}^{N_{2}} B_{H_{i}}^{1} + \ldots + \sum_{i=1}^{N_{2}} B_{H_{i}}^{M} = \sum_{i=1}^{N_{2}} (B_{H_{i}}^{1} + \ldots + B_{H_{i}}^{M})$$

$$= \sum_{i=1}^{N_{2}} \frac{C_{H_{i}}}{N_{1} + M - 1} = \frac{N_{2}}{N_{1} + M - 1} \overline{C_{H}}$$
(5.11)

For M = 1 in (5.9) the Mutualcast result in (3.6) is obtained as a special case. For M = 2 and $N_2 = 0$ the result in (5.7) is obtained.

In this scenario, each peer assigns a different portion of its upload capacity to each video, in order to maximize the overall throughput. Thus, the individual throughputs Θ_i cannot be uniquely computed, because several sets of optimal individual throughputs can be combined to achieve the maximum overall throughput in the system. For example, the maximum overall throughput of 500 kbps using two sources, can be achieved if the individual throughputs Θ_1 and Θ_2 are 300 kbps and 200 kbps, respectively. However, the same maximum overall throughput can be achieved if Θ_1 and Θ_2 are 216.65 kbps and 283.35 kbps, respectively. Both individual values for Θ_1 and Θ_2 aim to maximize the overall throughput, but each individual throughput Θ_i has a set of different optimal values. Furthermore, the upload capacity of the requesting peers could be exhausted by forwarding one video sequence only, while the other video sequences are forwarded by the rest of the sources. For this scenario of sources with joint rate allocation there are typically several solutions to assign the rate, which maximize the overall throughput can be obtained via linear programming. This solution is shown in Section 5.4.

5.2.2. Sources with Joint Rate Allocation for Same Rate Streams

Different sources may have different upload capacities during a video streaming session and the videos may be streamed at different rates. Therefore, a requesting peer may receive at the same time a video with high quality while a second video is received with low quality. In order to deal with this issue, the second scenario again jointly decides the rate allocation for all participating peers, but additionally enforces same rate streams.

A. Particular Case

The particular case considers two sources which distribute two different video sequences to N_1 requesting peers. The case assumes that both videos are distributed with same rate $\Theta_1 = \Theta_2$. The source upload to the content requesting peers R_i for redistribution from the source S_1 is

$$B_R^1 = \sum_{i=1}^{N_1} B_{R_i}^1 \tag{5.12}$$

while the source upload to the content requesting peers R_i for redistribution from the source S_2 is

$$B_R^2 = \sum_{i=1}^{N_1} B_{R_i}^2 \tag{5.13}$$

Assuming that $B_{R_i}^1 = B_{R_i}^2$, B_R is obtained similar to (5.4)

$$B_R = B_R^1 + B_R^2 = \sum_{i=1}^{N_1} (B_{R_i}^1 + B_{R_i}^2) = \sum_{i=1}^{N_1} 2B_{R_i}^1 = \sum_{i=1}^{N_1} \frac{C_{R_i}}{N_1} = \frac{1}{N_1} \sum_{i=1}^{N_1} C_{R_i} = \overline{C_R}$$
(5.14)

The distribution throughput for the two videos now becomes

$$\Theta_{1,2} = min(C_{S_1}, C_{S_2}) \quad for \quad min(C_{S_1}, C_{S_2}) \le \frac{B_R}{2}$$
(5.15)

If $min(C_{S_1}, C_{S_2}) > \frac{B_R}{2}$, then either source S_1 or source S_2 or both S_1 and S_2 are not exhausted and they have the residual upload capacities ΔS_1 and ΔS_2 .

$$\Delta S_1 = C_{S_1} - \frac{B_R}{2} \tag{5.16}$$

$$\Delta S_2 = C_{S_2} - \frac{B_R}{2} \tag{5.17}$$

In order to maximize the overall throughput, the system must exhaust the upload capacity of both sources. The exhaustion of the sources S_1 and S_2 is illustrated in Figure 5.3. Here,



Figure 5.3.: *Exhaustion of* ΔS_1 *and* ΔS_2

 B_D^1 represents the video block directly sent from the source S_1 to N_1 requesting peers R_i and the source S_2 . $B_{S_2}^1$ represents the video block sent by the source S_1 to the source S_2 , which forwards the block to all requesting peers R_i . B_D^2 represents the video block directly sent from the source S_2 to N_1 requesting peers and the source S_1 , and $B_{S_1}^2$ represents the video block sent by the source S_2 to the source S_1 , which forwards the block to all requesting peers R_i . The exhaustion of ΔS_1 and ΔS_2 is given by

with the same rate constraint $B_D^1 + B_{S_2}^1 = B_D^2 + B_{S_1}^2 = \Delta \Theta_1 = \Delta \Theta_2$. This leads to an optimization problem, which mathematically can be formulated as:

$$\begin{aligned} Maximize \quad \Delta\Theta_1 &= \Delta\Theta_2 = B_D^1 + B_{S_2}^1 = B_D^2 + B_{S_1}^2 \\ subject \ to \\ & (N_1 + 1)B_D^1 + N_1 B_{S_1}^2 + B_{S_2}^1 \le \Delta S_1 \\ & (N_1 + 1)B_D^2 + N_1 B_{S_2}^1 + B_{S_1}^2 \le \Delta S_2 \end{aligned}$$
(5.19)

The objective in (5.19) is to find the optimal rate allocation that maximizes the additional throughput $\Delta\Theta_1$ and $\Delta\Theta_2$, while enforcing the same rate for both videos. The linear optimization problem can be solved using efficient linear programming (LP) solvers [107], in a similar way as it is proposed in [106]. Then, the overall throughput Θ is

$$\Theta = \Theta_1 + \Theta_2 \tag{5.20}$$

where $\Theta_1 = \Theta_2 = \frac{B_R^1}{2} + \Delta \Theta_1 = \frac{B_R^2}{2} + \Delta \Theta_2$.

A special case for $\Delta \Theta_1$ and $\Delta \Theta_2$ is obtained when $\Delta S_1 > \Delta S_2$ and more specifically $\Delta S_1 \ge (2N_1 + 1)\Delta S_2$, then

$$\Delta \Theta_1 = \Delta \Theta_2 = \Delta S_2 \tag{5.21}$$

Similarly for $\Delta S_2 \ge (2N_1 + 1)\Delta S_1$,

$$\Delta \Theta_1 = \Delta \Theta_2 = \Delta S_1 \tag{5.22}$$

B. General Case

To extend the second scenario to its general case, M sources, N_1 requesting peers R_i , and N_2 helper peers H_i are considered. Again, helper peers just contribute their upload capacity during distribution and they are not interested in receiving the videos. The case assumes that the distribution throughput for all M sources is the same:

$$\Theta_1 = \ldots = \Theta_j = \ldots = \Theta_M \tag{5.23}$$

The source upload to the content requesting peers and the helper peers for redistribution from the source S_j is

$$\sum_{i=1}^{N_1} B_{R_i}^j + \sum_{i=1}^{N_2} B_{H_i}^j$$
(5.24)

Assuming that $B_{R_i}^1 = ... = B_{R_i}^j = ... = B_{R_i}^M$ and $B_{H_i}^1 = ... = B_{H_i}^j = ... = B_{H_i}^M$, then

$$B_R = \sum_{i=1}^{N_1} (B_{R_i}^1 + \dots + B_{R_i}^j + \dots + B_{R_i}^M) = \sum_{i=1}^{N_1} M * B_{R_i}^1 = \overline{C_R}$$
(5.25)

and

$$B_H = \sum_{i=1}^{N_2} (B_{H_i}^1 + \dots + B_{H_i}^j + \dots + B_{H_i}^M) = \sum_{i=1}^{N_2} M * B_{H_i}^1 = \overline{C_H}$$
(5.26)

Then, $\Theta_{1,...,M} = min(C_{S_1},...,C_{S_M})$ for $min(C_{S_1},...,C_{S_M}) \leq \frac{(B_R+B_H)}{M}$.

If $min(C_{S_1}, \ldots, C_{S_M}) > \frac{(B_R + B_H)}{M}$, then sources S_1, \ldots, S_M are not exhausted and they have a residual upload capacity ΔS_j

$$\Delta S_j = C_{S_j} - \frac{B_R + B_H}{M} \qquad j = 1, \dots, M$$
(5.27)

Then, the exhaustion of $\Delta S_1, \ldots, \Delta S_M$ is given by

$$(N_1 + M - 1)B_D^j + \sum_{i=1, i \neq j}^M B_{S_i}^j + (N_1 + M - 2)\sum_{i=1, i \neq j}^M B_{S_i}^j = \Delta S_j$$
(5.28)

67

with

$$B_D^1 + B_{S_2}^1 + \ldots + B_{S_M}^1 = \ldots = B_D^j + B_{S_1}^j + \ldots + B_{S_{j-1}}^j + B_{S_{j+1}}^j + \ldots + B_{S_M}^j$$

= \dots = B_D^M + B_{S_1}^M + \dots + B_{S_{M-1}}^M = \Delta \Theta_1 = \dots = \Delta \Theta_j = \dots = \Delta \Theta_M

 B_D^j represents the video block directly sent from the source S_j to all N_1 requesting peers R_i and the other M - 1 sources S_i . $B_{S_i}^j$ represents the video block sent by the source S_j to the source S_i , which forwards the block to N_1 requesting peers R_i and M - 2 sources S_i . $\Delta \Theta$ is the additional throughput, which becomes

$$\Delta \Theta = \sum_{i=1}^{M} \Delta \Theta_i \tag{5.29}$$

The overall throughput is then given as

$$\Theta = M * \Theta_i + \Delta \Theta \tag{5.30}$$

The problem can be transformed to the following equivalent optimization problem

$$\begin{aligned} Maximize & \Delta\Theta_1 = \ldots = \Delta\Theta_j = \ldots = \Delta\Theta_M \\ subject to \end{aligned} \tag{5.31}$$

$$(N_1 + M - 1)B_D^j + \sum_{i=1, i \neq j}^M B_{S_i}^j + (N_1 + M - 2)\sum_{i=1, i \neq j}^M B_{S_i}^j \le \Delta S_j$$
(5.32)

for all j = 1, ..., M.

The objective in (5.31) is to find the optimal rate allocation that maximizes the additional throughput $\Delta\Theta_1$ to $\Delta\Theta_M$, while enforcing the same rate for all videos. The constraints in (5.32) ensure that the limits on the upload capacity of the senders are not exceeded. This optimization problem can be solved efficiently using common optimization techniques such as the simplex method [107].

Similar to the first scenario, each peer has a wide freedom to assign a different portion of its upload capacity to each video, in order to achieve the same throughput for all video sequences. Thus, the individual throughputs Θ_i cannot be computed by a specific equation, and a set of optimal individual throughputs can be combined to achieve the same throughput for different video sequences.

5.2.3. Sources with Independent Rate Allocation for Same Rate Streams

The third scenario considers the case where the sources work separately with independent rate allocation and the same rate for all videos is enforced. This scenario also assumes that the sources and the requesting peers distribute their upload capacities in an equal way among all different video sequences. This means that coordination between the sources is not happening and excess capacity of a source cannot be exhausted by other sources.

A. Particular Case

For the particular case, two sources and N_1 requesting peers R_i are considered. The case assumes that two videos are distributed from the sources and that all requesting peers and

both sources need to receive all videos. The case also assumes that all peers distribute their upload capacities equally among all video sequences as shown in Figure 5.4. In other words, two separate and independent Mutualcast distributions run in parallel. Since same rate streams are delivered, the distribution throughput for both sources is the same, i.e, $\Theta_1 = \Theta_2$.



Figure 5.4.: An example of two sources with independent rate allocation and same rate streams

Figure 5.4 shows how the sources S_1 and S_2 divide their upload capacity in two parts for delivery of the videos X and Y. The source upload to the requesting peers for redistribution from the source S_1 and S_2 is given by (5.12) and (5.13), respectively.

Assuming that $B_{R_i}^1 = B_{R_i}^2$, and dividing the upload capacity of the requesting peers C_{R_i} by two, B_R is obtained similar to (5.4)

$$B_R = \sum_{i=1}^{N_1} (B_{R_i}^1 + B_{R_i}^2) = \sum_{i=1}^{N_1} 2 * B_{R_i}^1 = \frac{\sum_{i=1}^{N_1} C_{R_i}}{2 * N_1} = \frac{\overline{C_R}}{2}$$
(5.33)

If $\frac{C_{S_1}}{2} \leq B_R$, then the overall throughput Θ is limited by $\frac{C_{S_1}}{2}$. On the other hand, if $\frac{C_{S_1}}{2} > B_R$ the upload capacity of source S_1 allocated to video X is not exhausted and a data block $B_{S_2}^1$ defined as

$$B_{S_2}^1 = \frac{C_{S_2}}{2 * N_1} \tag{5.34}$$

is sent to the source S_2 for redistribution to N_1 requesting peers. Additionally, one block B_D^1 is directly sent to each requesting peer and source S_2 . Since both sources are independent, a coordination between both sources is not happening and the remaining capacity of a source cannot be exhausted by the other source. Thus, the residual upload capacity Δ in source S_2 cannot be used by the source S_1 and the source S_2 is not exhausted. Therefore, the individual maximum throughput for the first video sequence can be written as

$$\Theta_{1} = \begin{cases} \frac{C_{S_{1}}}{2} & \text{for } \frac{C_{S_{1}}}{2} \le B_{R} \\ B_{R} + B_{S_{2}}^{1} + \frac{\frac{C_{S_{1}}}{2} - (B_{R} + B_{S_{2}}^{1})}{N_{1} + 1} & \text{for } \frac{C_{S_{1}}}{2} > B_{R} \end{cases}$$
(5.35)

Similarly, the individual maximum throughput for the second video sequence can be written as

$$\Theta_{2} = \begin{cases} \frac{C_{S_{2}}}{2} & \text{for } \frac{C_{S_{2}}}{2} \le B_{R} \\ B_{R} + B_{S_{1}}^{2} + \frac{\frac{C_{S_{2}}}{2} - (B_{R} + B_{S_{1}}^{2})}{N_{1} + 1} & \text{for } \frac{C_{S_{2}}}{2} > B_{R} \end{cases}$$
(5.36)

with $B_{S_1}^2 = \frac{C_{S_1}}{2*N_1}$.

Similar to the second scenario, the third scenario enforces the same rate for all videos. In this case, the overall throughput becomes

$$\Theta = 2 * \min(\Theta_1, \Theta_2) \tag{5.37}$$

this means that the overall throughput is limited by the source with the smallest upload capacity.

B. General Case

To extend the third scenario with independent rate allocation to its general case, M sources S_i , N_1 requesting peers R_i and N_2 helper peers H_i are considered. The case assumes that M separate Mutualcast distributions run in parallel and the distribution rate for all the sources is the same $\Theta_1 = \ldots = \Theta_j = \ldots = \Theta_M$. The case also assumes that all peers distribute their upload capacities in an equal way among all different video sequences. The individual maximum throughput for each video can be written as

$$\Theta_{j} = \begin{cases} \frac{C_{S_{j}}}{M} & \text{for } \frac{C_{S_{j}}}{M} \le (B_{R} + B_{H} + B_{S}^{j}) \\ B_{R} + B_{H} + B_{S}^{j} + \frac{\frac{C_{S_{j}}}{M} - (B_{R} + B_{H} + B_{S}^{j})}{(N_{1} + M - 1)} & \text{for } \frac{C_{S_{j}}}{M} > B_{R} + B_{H} + B_{S}^{j} \end{cases}$$
(5.38)

with $B_S^j = \sum_{i=1, i \neq j}^M B_{S_i}^j$, and the contribution $B_{S_i}^j$ from the other sources S_i

$$B_{S_i}^j = \frac{C_{S_i}}{M(N_1 + M - 2)} \quad for \quad i = 1, \dots, M, \quad i \neq j$$
(5.39)

the contribution B_R from the requesting peers

$$B_R = \frac{\sum_{i=1}^{N_1} C_{R_i}}{M(N_1 + M - 2)} \tag{5.40}$$

and the contribution B_H from the helper peers

$$B_H = \frac{\sum_{i=1}^{N_2} C_{H_i}}{M(N_1 + M - 1)}$$
(5.41)

Similar to the second scenario the same rate for all videos is enforced. In this case, the overall throughput becomes

$$\Theta = M * min(\Theta_1, \dots, \Theta_M) \tag{5.42}$$

Again, the maximum overall throughput is limited by the source with the smallest upload capacity in the system. Or in other words, the source with the smallest value in (5.38) determines the rate of all videos. Typically, not all sources will be exhausted with respect to their upload capacity.

5.2.4. Sources with Independent Rate Allocation for Different Rate Streams

Similar to the third scenario, the fourth scenario considers the case where the sources work separately with independent rate allocation, but this time without enforcing the same rate for all videos.

A. Particular Case

The particular case considers two sources and N_1 requesting peers R_i . Two videos are distributed from the sources and all requesting peers and both sources need to receive all videos. The case also assumes that all peers distribute their upload capacities in an equal way among both video sequences, similar to two separate Mutualcast schemes running in parallel. However, contrary to the third scenario, here both video sequences can have different rates. In this case, the individual throughputs for the first video and second video are calculated by using (5.35) and (5.36), respectively. Since this scenario does not enforce same rate videos, Θ_1 and Θ_2 can be different and the sources can maximize the distribution of their videos exhausting their upload capacity completely. Thus, the overall throughput becomes

$$\Theta = \Theta_1 + \Theta_2 \tag{5.43}$$

Here, the overall throughput is the sum of both individual throughputs, because the constraint $\Theta_1 = \Theta_2$ has been removed. Thus, each source can exhaust half of its upload capacity assigned to its local video. However, since both sources are independent, a coordination between them does not exist and the remaining capacity of the second half of their upload capacity cannot be exhausted by the other source. This problem is present when both sources have different upload capacity.

B. General Case

To extend the fourth scenario with independent rate allocation to its general case, M sources S_i , N_1 requesting peers R_i and N_2 helper peers H_i are considered. The individual maximum throughput for each video can be computed by (5.38). Since this case does not enforce same rate videos, the individual throughputs can be different and the overall throughput becomes

$$\Theta = \sum_{j=1}^{M} \Theta_j \tag{5.44}$$

The sources can send different rate videos and each source can fully exhaust its upload capacity assigned to its local video. However, a portion of its upload capacity assigned to the other videos cannot be exhausted due to the lack of coordination among the sources.

5.3. PSNR-based Analysis

An alternative to throughput maximization is to maximize the overall end-to-end video quality for all delivered videos. In this section, end-to-end video quality maximization for all delivered videos is studied. The peak signal-to-noise ratio (PSNR) is used as a measure of video quality. In [104], two models have been developed for the sequence level rate-distortion (R-D) performance of predictive video source encoding. Both models require very limited amount of empirical data, namely three pairs of rate and distortion, in order to set up the model parameters. Experimental validations using H.264/AVC encoded video test sequences report high accuracy of these two proposed models. The R-D model from [104] that is adopted in this analysis relates the PSNR of a video sequence to the encoding rate R as

$$PSNR(R) = a + b\sqrt{\frac{R}{c}}(1 - \frac{c}{R})$$
(5.45)

where the parameter tuple (a, b, c) can be obtained by measuring three pairs of rate and distortion. The analysis uses H.264/AVC JM reference software version 12 [105] and selects three different quantization parameters to obtain three pairs of *PSNR* and rate for a specific video sequence. In the proposed framework the coding rate R_i for the video of source S_i is defined by the throughput Θ_i . Thus, the maximum overall *PSNR* is proportional to the maximum overall throughput reached. The overall end-to-end video quality of all delivered videos in the framework is given by

$$PSNR_{total} = \sum_{i=1}^{M} (PSNR_i)$$
(5.46)

where M is the number of distributed videos (sources) in the P2P network.

Similar as in the throughput-based analysis, four scenarios are considered for PSNR-based resource allocation:

- 1. Sources with joint rate allocation for different video quality streams,
- 2. Sources with joint rate allocation for same video quality streams,
- 3. Sources with independent rate allocation for same rate streams,
- 4. Sources with independent rate allocation for different video quality streams.

For each scenario a particular case and the general case are analyzed. The particular case considers two sources and N_1 requesting peers R_i , while the general case considers M sources S_i , N_1 requesting peers R_i and N_2 helper peers H_i . In the third and fourth scenario, the sources are independent, and the coordination among all sources in order to enforce the same PSNR for all videos is not possible. Therefore, in these scenarios, the analysis focuses on maximizing the video quality when the streams have same or different rate.

5.3.1. Sources with Joint Rate Allocation for Different Video Quality Streams

In the first scenario, the sum of *PSNR* for all video sequences is maximized without enforcing the same quality for all sequences.

A. Particular Case

The particular case assumes that each video has a different D - R function and hence a different PSNR at the same rate. The $PSNR_1$ for the first video sequence transmitted from source S_1 is

$$PSNR_1 = a_1 + b_1 \sqrt{\frac{R_1}{c_1}} (1 - \frac{c_1}{R_1})$$
(5.47)

where $R_1 = \Theta_1$.

The $PSNR_2$ for the the second video, which is transmitted from source S_2 is

$$PSNR_2 = a_2 + b_2 \sqrt{\frac{R_2}{c_2}} (1 - \frac{c_2}{R_2})$$
(5.48)

where $R_2 = \Theta_2$. The case assumes that in general $R_1 \neq R_2$.

Given the upload capacity exhaustion for source S_1 in (5.1), the upload capacity exhaustion for source S_2 in (5.2), and the upload contribution from the requesting peers R_i in (5.4), the maximization of the accumulated PSNR leads to an optimization problem, which can be expressed as

$$\begin{aligned} Maximize & \sum_{i=1}^{2} (PSNR_{i}) \\ subject \ to \end{aligned} \tag{5.49} \\ & \sum_{i=1}^{N_{1}} B_{R_{i}}^{1} + B_{S_{2}}^{1} + (N_{1}+1)B_{D}^{1} + N_{1}B_{S_{1}}^{2} \leq C_{S_{1}} \\ & \sum_{i=1}^{N_{1}} B_{R_{i}}^{2} + B_{S_{1}}^{2} + (N_{1}+1)B_{D}^{2} + N_{1}B_{S_{2}}^{1} \leq C_{S_{2}} \\ & \sum_{i=1}^{N_{1}} B_{R_{i}}^{2} + B_{S_{1}}^{2} + (N_{1}+1)B_{D}^{2} + N_{1}B_{S_{2}}^{1} \leq C_{S_{2}} \\ & B_{R_{1}}^{1} + \dots + B_{R_{1}}^{j} + \dots + B_{R_{1}}^{M} \leq C_{R_{1}} \\ & \vdots \\ & B_{R_{N_{1}}}^{1} + \dots + B_{R_{N_{1}}}^{j} + \dots + B_{R_{N_{1}}}^{M} \leq C_{R_{N_{1}}} \end{aligned}$$

The objective function in (5.49) maximizes the sum of the individual PSNR values. The first and second constraints in (5.50) ensure that the limit of the capacity of the sources S_1 and S_2 are not exceeded, while the last N_1 constraints ensure that the number of bits redistributed by the requesting peers R_1 to R_{N_1} do not exceed their upload capacities. This optimization problem can be solved using linear programming.

B. General Case

To extend the first PSNR scenario to its general case, M sources S_i , N_1 requesting peers R_i and N_2 helper peers H_i are considered. The case assumes that each video has a different PSNR versus rate function

$$PSNR_{1} = a_{1} + b_{1}\sqrt{\frac{R_{1}}{c_{1}}}(1 - \frac{c_{1}}{R_{1}})$$

$$\vdots \qquad \vdots$$

$$PSNR_{j} = a_{j} + b_{j}\sqrt{\frac{R_{j}}{c_{j}}}(1 - \frac{c_{j}}{R_{j}})$$

$$\vdots \qquad \vdots$$

$$PSNR_{M} = a_{M} + b_{M}\sqrt{\frac{R_{M}}{c_{M}}}(1 - \frac{c_{M}}{R_{M}})$$
(5.51)

The upload capacity exhaustion requirement for the sources S_1 to S_M is given by (5.8), while the upload contribution from the requesting peers R_i is given by (5.10) and the upload capacity contribution from the helper peers H_i is given by (5.11). Mathematically, the overall PSNR maximization can be formulated as

$$\begin{aligned} Maximize & \sum_{i=1}^{M} (PSNR_i) \\ subject \ to \\ & \sum_{i=1}^{N_1} B_{R_i}^j + \sum_{i=1}^{N_2} B_{H_i}^j + \sum_{i=1, i \neq j}^{M} B_{S_i}^j + (N_1 + (M-1)) B_D^j \le C_{S_j} \\ for \ all \ j = 1, \dots, M. \\ & \sum_{i=1}^{M} B_{R_1}^i \le C_{R_1}; \dots; \sum_{i=1}^{M} B_{R_{N_1}}^i \le C_{R_{N_1}} \\ & \sum_{i=1}^{M} B_{H_1}^i \le C_{H_1}; \dots; \sum_{i=1}^{M} B_{H_{N_2}}^i \le C_{H_{N_2}} \end{aligned}$$
(5.53)

The objective function in (5.52) maximizes the sum of PSNR for all videos. The constraints (5.53) ensure that the limit of the upload capacities of the sources, the requesting peers and the helper peers are not exceeded. Again, linear programming is used to solve this optimization problem.

5.3.2. Sources with Joint Rate Allocation for Same Video Quality Streams

The second scenario assumes that the sources jointly decide the rate allocation for all participating peers, but additionally enforces the same video quality for all video streams.

A. Particular Case

The particular case considers two sources (S_1 and S_2) distributing two different video sequences with same *PSNR*. Initially, the rates R_i generated by both sources are assumed to be identical. The upload requirement for source S_1 is then given by (5.1), while the total upload requirement for source S_2 is given by (5.2). The upload contribution from the requesting peers is given by (5.4). The individual PSNRs for the videos 1 and 2 are given by (5.47) and (5.48), respectively. Here, $PSNR_1$ and $PSNR_2$ are computed using the rates R_1 and R_2 , respectively. The sum of PSNRs is given by (5.46), where M is set to 2. In this case, the overall PSNR is maximized for two different video sequences using (5.49), but enforcing the same rate for both videos ($R_1 = R_2$). The enforcement can be realized by adding the constraint $R_1 = R_2$ in (5.50). The same quality for all videos is obtained if they have the same RD-function. However, when the same rate for all video sequences is not enough to obtain a similar video quality among them, a same PSNR need to be enforced. The PSNRenforcement is possible, when the sources have abundant upload capacity. To this end, the broadcast links in each source are manipulated, and the rate of the sequence with the largest rate is reduced using a rate allocation mechanism. In the following, a description of this adaptive mechanism to control the rate allocation between the sources is given.

The goal of the adaptive rate control scheme is to effectively use the available upload capacity from each source to deliver a homogeneous video quality for all streams. To this end, each source schedules the distribution according to the ratio of the video bit rates. The mechanism assumes that the quality requirements are known. Then, a number of layers to reach this quality level are determined for each video in each source using scalable video coding. Determining the number of layers and the coding rate for two different video sequences is illustrated in Figure 5.5.



Figure 5.5.: Enforcement of the same video quality for two different videos using scalable video coding. *a*). Redistribution of layers, *b*). PSNR comparison

In Figure 5.5 a), the sources send the base layer of their videos in Case 1. In Case 2, both sources send three enhancement layers of their videos, and the rate R_4 for both sequences is the same. The example assumes that video 1 and video 2 are different and they have been encoded with different bit rates. Thus, using the rate R_4 for both videos, a *PSNR* of 45 dB and 38 dB for video 1 and video 2, are obtained respectively. In order to enforce the same *PSNR* for both sequences scalable video coding is used in Case 3. Then, the source 1 sends

two enhancement layer of video 1, while the source 2 sends four enhancement layer of video 2. Figure 5.5 b) shows how both videos sequences can reach the same *PSNR* using different number of enhancement layers and different rate.

Once the number of required layers and the coding rate are known in each source and before starting the distribution, the sources exchange the coding rate of their videos. Each source computes a local distribution ratio k using these coding rates. The case considers M = 2. This ratio is used in each source to determine the number of required packets for each video. In an ideal situation it is desirable that the throughput is the same as the playback bit rate of the videos in order to obtain a short initial waiting time and a minimal size of buffers. In contrast, if the upload capacity of the sources is not enough to satisfy the requested throughput, the initial time and the buffer usage is increased. Additionally, when different videos are distributed from different sources, the sources need to synchronize the playback of the videos and adapt their upload allocation for the distribution, so that all videos can be received with adaptive throughput and have similar initial waiting time or video quality. The sources use the distribution ratio to adapt the distribution throughput of streams and each source can schedule the number of packets to distribute from itself and the number of packet to distribute from the other sources. Thus, the number of distributed packets for each video is proportional to its coding rate. In the proposed mechanism, the distribution ratios are fixed, however, the amount of distributed packets is variable according to the available upload capacity of the sources or the network performance. The distribution ratio is used to manipulate all delivery links in each source.

The rate control mechanism using two sources (S_1 and S_2) and two requesting peers (R_1 and R_2) is shown in Figure 5.6. Here, sources S_1 and S_2 distribute video X and video Y, respectively. Video X and Y have a coding rate R_1 and R_2 , respectively.



Figure 5.6.: Adaptive rate control scheme for two sources

The distribution ratios for the sources S_1 and S_2 are computed as

$$k_{S_1} = \frac{R_2}{R_1} \tag{5.54}$$

and

$$k_{S_2} = \frac{R_1}{R_2},\tag{5.55}$$

respectively.

Each source distributes packets of its local video and redistributes the packets received from the other source. Thus, source S_1 distributes n_1 packets of video X to all peers, including the source S_2 , while redistributing q_1 packets of video Y from S_2 to each requesting peer. Source S_1 uses the ratio k_{S1} to obtain a distribution proportion between q_1 and n_1 such that

$$q_1 = \lceil n_1 * k_{S_1} \rceil \tag{5.56}$$

Source S_2 distributes q_2 packets of video Y to all peers, including the source S_1 , while redistributing n_2 packets of video X from S_1 to each requesting peer. Source S_2 uses the ratio k_{S2} to obtain a distribution proportion between n_2 and q_2 , such that

$$n_2 = \lceil q_2 * k_{S_2} \rceil \tag{5.57}$$

B. General Case

To extend the analysis to M sources with N_1 requesting peers and N_2 helper peers, the general case assumes that the total upload requirement for sources S_1 to S_M is given by (5.8), the upload capacity contribution from requesting peers is given by (5.10) and the upload capacity contribution from the helper peers is given by (5.11). Each video sequence generated from each source shows a different PSNR versus rate function (see (5.51)). However, in this scenario all rates R_i for all M sources are identical. Thus, R_1 to R_M can be computed by dividing the maximum overall throughput Θ between M. $PSNR_1$ to $PSNR_M$ are calculated using the values of R_1 to R_M respectively. The sum of $PSNR_S$ is given by (5.46). Similar to the particular case, when the same rate for all video sequences is not enough to obtain a similar video quality among them, a same PSNR need to be enforced using the rate allocation mechanism. For multiple source, each source computes different distribution ratios for its local video sequences and the different video sequences to be redistributed from other sources. This leads to a combinatorial optimization problem.

5.3.3. Sources with Independent Rate Allocation for Same Rate Streams

The third scenario considers the case where the sources work separately with independent rate allocation and same rate for all videos. The third scenario assumes that the sources distribute their upload capacities equally among all video sequences. The scenario also assumes that all requesting peers and helper peers divide their upload capacity equally among all video sequences. Since all sources are independent, the coordination among the sources to enforce the same *PSNR* for all videos is not possible. However, the same rate can be enforced, which would lead to similar video quality for similar video sequences.

A. Particular Case

The particular case considers two sources (S_1 and S_2) distributing two different video sequences and assumes that the rates R_i generated by both sources are identical. The particular case assumes that the upload capacity of the sources and the requesting peers is divided by two, in order to assign the same upload capacity for both videos. Sources S_1 and S_2 distribute two different video sequences with different *PSNR*. Since both sources distribute both videos with the same rate R_i , the case assumes that the rate R_i for both sources is limited by the capacity of the smallest source in the system. Thus,

$$R_1 = R_2 = \frac{\Theta}{2} \tag{5.58}$$

where Θ is given by (5.37).

The PSNRs for video sequence 1 and 2, are given by (5.47) and (5.48), respectively, and the total PSNR is computed by (5.46), where M is set to 2. Although the coordination between both sources is not possible, enforcing same rate leads to similar PSNR for similar videos.

B. General Case

To extend the third scenario to its general form, M sources with N_1 requesting peers and N_2 helper peers are considered. The case assumes that all rates R_i for all M sources are identical and all participating peers equally divide their upload capacity among all videos.

Since all sources distribute all videos with the same rate R_i , the rate R_i for all sources is limited by the capacity of the weakest source in the system. Thus,

$$R_1 = \ldots = R_j = \ldots = R_M = \frac{\Theta}{M}$$
(5.59)

where Θ_i is given by (5.42).

Each video sequence generated from each source displays a different PSNR, which is computed by (5.51), and the sum of PSNRs is computed by (5.46). Similar to the particular case, if all distributed videos have approximately the same D - R function, the same rate R_i for all videos can be enforced in order to obtain similar video quality.

5.3.4. Sources with Independent Rate Allocation for Different Video Quality Streams

The fourth scenario considers the case where the sources work separately with independent rate allocation and different rate streams. In other words, the scenario does not enforce the same rate for all videos, and assumes that all sources distribute their upload capacities equally among all video sequences. The scenario also assumes that all requesting peers and helper peers divide their upload capacity equally among all sequences. Coordination among the sources to enforce the same *PSNR* for all videos is not possible. In other words, the remaining capacities of a source cannot be exhausted by the other sources.

A. Particular case

The particular case considers two sources (S_1 and S_2) distributing two different video sequences to N_1 requesting peers. The particular case assumes that the rate R_i for the videos are different, and that the upload capacity of the sources and the requesting peers is divided by two, in order to assign the same upload capacity for both videos.

The individual *PSNR* for videos 1 and 2 is computed by (5.47) and (5.48), respectively, where $R_1 = \Theta_1$ and $R_2 = \Theta_2$. Throughput Θ_1 and Θ_2 is computed by (5.35) and (5.36), respectively. The total overall *PSNR* represents the sum of the individual *PSNR*s, which is

given by (5.46), where *M* is set to 2.

B. General case

To extend the fourth scenario to its general form, M sources with N_1 requesting peers and N_2 helper peers are considered. The case assumes that the sources, the requesting peers and helper peers equally divide their upload capacity among all transmitted videos.

Each video sequence generated from each source displays a different *PSNR*, which is computed by (5.51). The individual rates R_i for each video are given by the individual throughput Θ_i , where the individual throughput Θ_i for each video sequences is computed by (5.38). Then, $R_1 = \Theta_1, \ldots, R_j = \Theta_j, \ldots, R_M = \Theta_M$. The total *PSNR* represents the sum of all individual *PSNRs*, and it is computed by (5.46).

5.4. Simulation

5.4.1. Throughput-based Simulation

In this section, the proposed framework is evaluated using linear programming. The first part of the simulation evaluates the framework in terms of overall throughput, and linear programming is used to maximize the overall throughput. In order to simulate the analytical framework presented in Section 5.2, the following scenarios are considered:

- 1. Sources with joint rate allocation for different rate streams,
- 2. Sources with joint rate allocation for same rate streams,
- 3. Sources with independent rate allocation for same rate streams,
- 4. Sources with independent rate allocation for different rate streams.

For all cases, a set of 5 participating peers are used, where two peers S_1 and S_2 act as sources and three peers R_1 , R_2 and R_3 are requesting peers. The upload capacity of the participating peers C_{S_1} , C_{S_2} , C_{R_1} , C_{R_2} and C_{R_3} in kbps is 300, 600, 500, 300, and 300, respectively. Helper peers are not considered in this example.

In the first scenario, the rate allocation is jointly decided for all participating sources and requesting peers. In order to maximize the overall throughput, this scenario considers that video sequences X and Y can have a different rate. The assumptions are translated into a linear program which is shown in Figure 5.7. Here, each source contributes its upload capacity not only to distribute its original content, but also the blocks received from the other source.

The first constraint $X_1 + X_2 + X_3 + X_4 + 4X_5 + 3Y_4 \le 300$ represents the upload capacity of source S_1 , which has to deliver the blocks X_1 , X_2 and X_3 to the requesting peers R_1 , R_2 , and R_3 , respectively. Source S_1 exhausts its upload capacity by sending a block X_4 to source S_2 for redistribution to all requesting peers, by delivering a block X_5 directly to each requesting peer and source S_2 and by redistributing block Y_4 received from source S_2 to each requesting peer. Each requesting peer receives one block of X and Y from the sources S_1 and S_2 , and the upload capacity of each requesting peer is exhausted by redistributing the blocks to the rest of the requesting peers and one source. Similar to the first constraint in Figure 5.7, the second constraint represents how the upload capacity of the source S_2 is exhausted. The

$$\begin{split} & \text{Maximize} \\ & \varTheta{\Theta} = X_1 + X_2 + X_3 + X_4 + X_5 + Y_1 + Y_2 + Y_3 + Y_4 + Y_5 \\ & \text{subject to} \\ & X_1 + X_2 + X_3 + X_4 + 4X_5 + 3Y_4 \leq 300 \,, \\ & Y_1 + Y_2 + Y_3 + Y_4 + 4Y_5 + 3X_4 \leq 600 \,, \\ & 3X_1 + 3Y_1 \leq 500 , \\ & 3X_2 + 3Y_2 \leq 300 , \\ & 3X_2 + 3Y_2 \leq 300 , \\ & 3X_3 + 3Y_3 \leq 300 , \\ & 0 \leq X_1, \ 0 \leq Y_1, \ 0 \leq X_2, \ 0 \leq Y_2, \ 0 \leq X_3, \ 0 \leq Y_3, \\ & 0 \leq X_4, \ 0 \leq Y_4, \ 0 \leq X_5, \ 0 \leq Y_5, \end{split}$$

Figure 5.7.: Linear program for the first scenario. The objective is to maximize the overall throughput

third, fourth and fifth constraints represent the contribution of the requesting peers R_1 , R_2 and R_3 , respectively. The constraints $0 \le X_1$ to $0 \le Y_5$, mean that negative block sizes are not allowed. Here, if the block size is zero, it means that no block is transmitted on this link. To solve the linear program, the Mathematica 5.0 software packet is used. The solution gives a maximum throughput of 500 kbps, while the rate of the blocks in kbps is $X_1 = 83.33$, $X_2 = 50$, $X_3 = 50$, $X_4 = 116.67$, $X_5 = 0$, $Y_1 = 83.33$, $Y_2 = 50$, $Y_3 = 50$, $Y_4 = 0$, and $Y_5 = 16.67$, respectively. The rate of the sequences X and Y are 300 kbps and 200 kbps, respectively. In this case, the best overall throughput can be achieved and the upload capacity of both sources and all requesting peers is fully exhausted. However, the video sequences X and Yhave different rate and hence most likely different quality.

In the second scenario, the rate allocation is also decided jointly for all participating peers, but additionally, the same rate for video sequences X and Y is enforced. The resulting linear program is similar to the previous case, except that now the linear program assumes that $X_1 + X_2 + X_3 + X_4 + X_5 = Y_1 + Y_2 + Y_3 + Y_4 + Y_5$. The solution gives again a maximum throughput of 500 kbps, while the rate of the blocks in kbps is $X_1 = 83.33$, $X_2 = 50$, $X_3 = 50$, $X_4 = 66.66$, $X_5 = 0$, $Y_1 = 83.33$, $Y_2 = 50$, $Y_3 = 50$, $Y_4 = 16.66$ and $Y_5 = 50$, respectively. Now, the rate of the two sequences X and Y is 250 kbps and hence balanced. In this case, the best overall throughput can be achieved, the upload capacity of both sources and all requesting peers is fully exhausted, and the sequences X and Y have the same rate.

The third scenario considers independent rate allocation for every source. The sources equally divide their upload capacity between videos X and Y. Thus, for the delivery of each sequence, the upload capacity in each source is divided by two. Each requesting peer also divides its upload capacity equally between both sequences. Additionally, the scenario assumes that both videos have the same rate. The assumptions translated to a linear program are similar the assumptions shown in Figure 5.7, but now all upload capacities are divided by 2. For example, the upload capacity of source S_1 , which is 300 kbps, is represented as $X_1 + X_2 + X_3 + X_4 + 4X_5 \leq 300 * 0.5$ and $3Y_4 \leq 300 * 0.5$. Additionally, the constraint $X_1 + X_2 + X_3 + X_4 + 4X_5 = Y_1 + Y_2 + Y_3 + Y_4 + Y_5$, which enforces the same rate between both videos is added. The solution after solving the resulting linear program gives a maximum throughput of 300 kbps, while the rate of the blocks in kbps is $X_1 = 83.33$, $X_2 = 50$, $X_3 = 16.66$, $X_4 = 0$, $X_5 = 0$, $Y_1 = 83.33$, $Y_2 = 50$, $Y_3 = 16.66$, $Y_4 = 0$, $Y_5 = 0$, respectively. The size of the sequences X and Y are balanced with 150 kbps, but the maximum throughput is

smaller than for the first and second scenario.

The fourth scenario considers independent rate allocation for every source, but this time the videos can have different rates. The fourth scenario assumes that the sources divide their upload capacity equally between videos X and Y. Thus, for the delivery of each sequence, the upload capacity in each source is divided by two. Each requesting peer also divides its upload capacity equally between both video sequences. The assumptions translated to a linear program are similar to Figure 5.7, but all upload capacities are divided by 2. For example, the upload capacity of source S_1 , which is 300 kbps, is represented as $X_1 + X_2 +$ $X_3 + X_4 + 4X_5 \le 300 * 0.5$ and $3Y_4 \le 300 * 0.5$. In this case, no constraint to enforce the same rate between both videos is considered. The solution gives a maximum throughput of 400 kbps, while the rate of the blocks in kbps is $X_1 = 83.33$, $X_2 = 50$, $X_3 = 16.67$, $X_4 = 0$, $X_5 = 0$, $Y_1 = 83.33, Y_2 = 50, Y_3 = 50, Y_4 = 50$, and $Y_5 = 16.67$, respectively. The rate of the sequences X and Y are 150 kbps and 250 kbps, respectively. The fourth scenario presents a better overall throughput than the third scenario with independent sources for same rate streams. This is because in the fourth scenario, the sources uses different rate streams, and they can carry out a best allocation of their upload capacity. However, the overall throughput in this scenario is smaller than the overall throughput obtained by the first and second scenarios. This is because both sources are independent, and the coordination between them is not possible. Thus, the remaining capacities of a source cannot be exhausted by the other source.

To compare how the four simulated scenarios works in the multi-source video multicast framework, the simulations are extended by varying the upload capacity of the source S_1 from 200 to 900 kbps, while the upload capacity of the source S_2 and the requesting peers is maintained. Figure 5.8 shows these results.



Scenario 1: Sources with joint rate allocation for different rate streams Scenario 2: Sources with joint rate allocation for same rate streams Scenario 3: Sources with independent rate allocation for same rate streams Scenario 4: Sources with independent rate allocation for different rate streams

Figure 5.8.: Overall throughput comparison for all cases

The results show that the sources with joint rate allocation for same rate streams achieves the same maximum overall throughput as the sources with joint rate allocation for different rate streams, when an optimal bandwidth allocation is found. Contrary, the sources with independent rate allocation achieves an overall throughput smaller than the sources with joint rate allocation. In the independent sources with same rate streams, the throughput is limited by the upload capacity of the weakest source. On the other hand, since both sources are independent, the coordination between them is not possible and the remaining capacities of a source cannot be exhausted by other sources. However, one can also see that all approaches lead to identical throughput when both sources have the same upload capacity. Appendix A contains additional information about the different experiments conduced for these scenarios. The results obtained in the different simulations confirm the obtained results using the set of equations given in the throughput-based analytical framework in Section 5.2.

5.4.2. PSNR-based Simulation

The second part of the simulation evaluates the framework in terms of aggregate video quality. To measure video quality, the peak signal-to-noise ratio (*PSNR*) is used. *PSNR* is the most widely used objective video quality metric. A video sequence with a *PSNR* value between 30 and 40 dB usually is acceptable, while a video with a *PSNR* below 30 dB is typically quite bad [93]. To evaluate the framework in terms of video quality, the model is fitted in (5.45) for each video sequence. Here, the test sequences 'Foreman' and 'Mother and Daughter' (M & D) are used. The three R-D pairs used to fit the model in (5.45) are shown in Table 5.1.

		2 1	
Sequences	PSNR/rate (dB/kbps)	PSNR/rate (dB/kbps)	PSNR/rate (dB/kbps)
Foreman	29.41/42	41.23/510	45.7/1017
M & D	35.28/42	45.03/510	48.81/1017

Table 5.1.: PSNR and rate for video sequences

The R-D pairs shown in Table 5.1 are used to calculate the corresponding coefficients a, b and c, which are used to parametrize the R-D function. The resulting R-D models for the two test sequences are as follows:

Foreman: (a = 36.44, b = 4.75 and c = 197.97)

$$PSNR(R_1) = 36.44 + 4.75\sqrt{\frac{R_1}{197.97}}\left(1 - \frac{197.97}{R_1}\right)$$
(5.60)

Mother and Daughter: (a = 44.34, b = 4.05 and c = 425.18)

$$PSNR(R_2) = 44.34 + 4.05\sqrt{\frac{R_2}{425.18}}\left(1 - \frac{425.18}{R_2}\right)$$
(5.61)

Similar to the throughput-based simulation, the PSNR-based analytical framework is simulated for four different scenarios:

- 1. Sources with joint rate allocation for different video quality streams,
- 2. Sources with joint rate allocation for same video quality streams,
- 3. Sources with independent rate allocation for same rate streams,

4. Sources with independent rate allocation for different video quality streams.

A set of 5 participating peers is considered, where the two peers S_1 and S_2 act as sources while the three peers R_1 , R_2 and R_3 are requesting peers. To maximize the sum of PSNRof both video sequences, the same upload capacity settings for each participating peer as in the previous section is assumed. The system performance based on overall video quality is defined by (5.46), where M is set to 2.

The first scenario maximizes the sum of *PSNR*, while the *PSNR* for both videos can be different. The assumptions are translated into the linear program shown in Figure 5.9.

$$\begin{split} PSNR_{1} &= 36.44 + 4.75*Sqrt[(R_{1}/197.97)]*(1 - 197.97/R_{1});\\ PSNR_{2} &= 44.34 + 4.05*Sqrt[(R_{2}/425.18)]*(1 - 425.18/R_{2});\\ R_{1} &= X_{1} + X_{2} + X_{3} + X_{4} + X_{5};\\ R_{2} &= Y_{1} + Y_{2} + Y_{3} + Y_{4} + Y_{5};\\ Maximize:\\ PSNR_{1} + PSNR_{2}\\ subject to:\\ X_{1} + X_{2} + X_{3} + X_{4} + 4X_{5} + 3Y_{4} \leq 300,\\ Y_{1} + Y_{2} + Y_{3} + Y_{4} + 4Y_{5} + 3X_{4} \leq 600,\\ 3X_{1} + 3Y_{1} \leq 500,\\ 3X_{2} + 3Y_{2} \leq 300,\\ 3X_{3} + 3Y_{3} \leq 300,\\ 0 \leq X_{1}, 0 \leq Y_{1}, 0 \leq X_{2}, 0 \leq Y_{2}, 0 \leq X_{3}, 0 \leq Y_{3}, 0 \leq X_{4}, 0 \leq Y_{4}, 0 \leq X_{5}, 0 \leq Y_{5} \end{split}$$

Figure 5.9.: The first scenario is translated into a linear program for the maximization of the aggregate video quality

The set of constraints defined in the linear program are the same as in Figure 5.7. The linear program is solved using Mathematica 5.0. The solution maximizes the sum of *PSNR* to be 79.75 dB, while the rates of the blocks in kbps are $X_1 = 83.33$, $X_2 = 50$, $X_3 = 50$, $X_4 = 75.53$, $X_5 = 6.68$, $Y_1 = 83.33$, $Y_2 = 50$, $Y_3 = 50$, $Y_4 = 4.79$ and $Y_5 = 46.31$, respectively. The rate of the sequences *X* and *Y* are 265.5 kbps and 234.5 kbps, respectively. The *PSNR* values of the first and second sequence in dB are 37.85 and 41.90, respectively. The upload capacity of all peers is exhausted and the video quality of the two streams differs significantly.

The second scenario assumes that the sequences generated in each source have the same PSNR. Initially, the scenario uses the results obtained when the rates of the two sequences X and Y are identical. The assumptions are translated into a linear program as is illustrated in Figure 5.9, but adding the constraint $X_1 + X_2 + X_3 + 4 + X_5 = Y_1 + Y_2 + Y_3 + Y_4 + Y_5$. Thus, the description of the linear program for the interconnected sources with balanced video sequences is similar to the previous case, except that sequences X and Y have the same rate. The solution leads to a sum of PSNR of 79.71 dB, while the rates in kbps are $X_1 = 83.33$, $X_2 = 50$, $X_3 = 50$, $X_4 = 57.18$, $X_5 = 9.47$, $Y_1 = 83.33$, $Y_2 = 50$, $Y_3 = 50$, $Y_4 = 7.18$, $Y_5 = 59.47$ respectively. The rate for both sequences is 250 kbps, and the PSNR values of the first and second sequence in dB are 37.55 and 42.16, respectively. The resulting overall reception quality is close to the optimal quality PSNR obtained by the multi-source scheme with different video quality, which represents the best overall PSNR in the framework,

however the individual video quality for both videos is different. Then, the same PSNR for both sequences needs to be enforced. To this end, source S_2 doesn't send block Y_4 to source S_1 , to avoid exhausting the upload capacity of S_1 during the distribution of this block. Also, block Y_5 is not sent by the source S_2 in order to use this upload capacity to send the block X_4 to all requesting peers. After this, the PSNR values of the first and second sequence in dB are 38.44 and 41.21, respectively. The sum of PSNR is 79.66 dB. This result shows that the overall quality is reduced, but both sequences tend to reach more similar PSNR. The first sequence cannot reach the same PSNR as the second sequence, because the upload capacity of S_1 has been exhausted and the rate cannot be further increased. If exactly the same video quality is to be obtained, the rate of the second sequence would now have to be reduced which, however, would lead to a waste of available resources as the upload capacities would not be fully exhausted. Thus, the same PSNR for all video sequences can be only achieved when the sources have enough upload capacity.

The third scenario evaluates the overall video quality when the sources have independent rate allocation and the rate for both videos is the same. This scenario assumes that the sources distribute their upload capacities equally for videos X and Y. We also assume that each requesting peer divides its upload capacity equally to both videos, and translate all constraints into a linear program. When solving the corresponding linear program, a sum of PSNR of 75.05 dB is obtained. The rate of the blocks in kbps is $X_1 = 21.53$, $X_2 = 34.46$, $X_3 = 8.98$, $X_4 = 85.01$, $X_5 = 0$, $Y_1 = 59.36$, $Y_2 = 44.73$, $Y_3 = 14.75$, $Y_4 = 31.07$ and $Y_5 = 0.068$, respectively. The rate of the sequences X and Y is balanced with 150 kbps and the first and second sequence have a PSNR of 35.12 dB and 39.93 dB, respectively, but the aggregate PSNR is smaller when compared to the previous cases.

The fourth scenario evaluates the overall video quality when the sources work with independent rate allocation and the PSNR for both videos are different. Similar to the third scenario, the sources distribute their upload capacities equally for videos X and Y. The scenario also assumes that each requesting peer divides its upload capacity equally between both videos, but the same PSNR for both videos does not need to be enforced in this scenario. The assumptions translated to a linear program are similar to Figure 5.9, but the upload capacity of the sources and requesting peers is divided by 2. When solving the corresponding linear program, a sum of PSNR of 75.05 dB is obtained. The rate of the sequences X and Y is 150 kbps and 200 kbps, respectively. The first and second videos have a PSNR of 35.14 dB and 42.16 dB, respectively, while the total PSNR is 77.30 dB. The fourth scenario presents better PSNRs than the third scenario. However, the PSNR is smaller than the PSNR obtained in the first and second scenarios.

Similar to the throughput-based comparison, the PSNR-based model is compared for the four introduced scenarios for a set of five participating peers. The upload capacity of the participating peers are the same as in the throughput-based comparison. The upload capacity of the source S_1 is varied from 200 to 900 kbps, while the upload capacity of the source S_2 and the requesting peers is maintained. Figure 5.10a compares the overall video quality for the evaluated scenarios. Here, the overall *PSNR* for the multi-source scheme with same video quality is shown after the *PSNR* enforcement has been made. The results show that when the source S_1 is weak (e.g., 200 kbps), the *PSNR* enforcement is not possible, because the maximum rate reached by sequence X is identical to the upload capacity of S_1 . The results also show that the multi-source scheme with same video quality close to the multi-source scheme with different video quality, but not equally.
This is because the first scheme redistributes its video quality in a similar PSNR between sequences X and Y. On the other hand, the multi-source scheme with independent rate allocation shows a smaller sum of PSNR, because the video quality is limited by the upload capacity of the weaker source. However, all approaches lead to similar overall PSNR when both sources have the same upload capacity.



Figure 5.10.: a). Overall PSNR comparison for all scenarios, *b). Individual PSNR comparison for all scenarios*

Scenario 4: Sources with independent rate allocation for different video quality streams

Figure 5.10b compares the individual *PSNR* of each video sequence for each scenario. The individual *PSNRs* obtained from the multi-source scheme with same video quality present the best video quality balance for both video sequences. In contrast, the individual *PSNRs* obtained for the multi-source scheme with independent rate allocation are less balanced. For this specific example, the *PSNR* enforcement is required because the video sequences are very different, however, when the video sequences from different sources are similar a strategy based on same rate may be enough to obtain a similar video quality for all video sequences in all participating peers.

The results obtained in the *PSNR* simulations agree with the obtained results using the set of equations given in the *PSNR*-based analytical framework. Appendix A contains additional information about the different experiments conduced for these scenarios.

5.5. Implementation and Evaluation

5.5.1. Implementation

A prototype collaborative multi-source multicast has been implemented in order to evaluate its performance in real world scenarios. The implementation runs on Linux and consists of different programs written in the C/C++ language. The implementation includes a server

module run by the source peers and a receiver module run by each requesting peer. Both modules have been enabled with a sender/receiver mode. Similar to MutualCast, in the implementation all links among the participating peers are established using TCP connections. Desired characteristics such as reliable data delivery, flow-control and handling of node leave events are automatically been taken care of by the TCP protocol. In the implemented prototype, all the participating peers receive all videos. This includes one source receiving the videos from other sources. In each source, the server module distributes the local video, while redistributing a part of video received from other sources, and receives other parts of videos from the requesting peers for its playback.

Each requesting peer runs a receiver module which receives the video blocks from the sources for its playback and forwards these blocks to the rest of the requesting peers and the sources that need to receive this content. In this collaborative way, all participating peers are sending and receiving at the same time. The distribution of blocks among the participating peers is implemented using threads. Threads are used in order to ensure distribution and storage without delay. Each source peer sets up the following threads:

- a sending thread to distribute own content to the requesting peers,
- a receiving thread to receive content from the sources and forward the data to the other peers (source is acting as receiving peer),
- a second receiving thread to receive the forwarded blocks form the other peers,
- a storing thread to read the blocks from storage,
- a measuring thread to perform the different measurements during the streaming session.

On the other hand, each requesting peer has the following threads:

- a receiving thread to receive data from the sources and forward the data to the other peers
- a second receiving thread to receive the forwarded blocks form the other peers,
- a storing thread to read the blocks from storage,
- a measuring thread to made the different measurements during the streaming session.

During a streaming session, the sending thread, receiving thread and measuring thread are concurrently running in each peer. To run the prototype, initially, the source starts listening on a pre-defined port and waits for the socket connection request. Then, each requesting peer establishes a connection with the source. The source copies the IP address and listening port of every requesting peer and sends this information to all requesting peers. After this, each peer starts a new thread, which listens and waits to establish the forward link for transferring content between two peers. All peers maintain a list of all peers in the multicast group. So the forwarding connection could be established very quickly. After the initialization of the forward link, all links are ready for content distribution. Then, the sources and the requesting peers begin the data transfer. After the blocks are received at each requesting peer or in each source, the blocks are forwarded to the rest of the participating peers. During the implementation, the block size is set to 1 KB, so each block can be sent using a single TCP/IP packet.

The distribution of the different videos is concurrently realized in each source by blocking and unblocking its distribution and redistribution queues until the videos have been distributed. Because the mechanism calculates the ratio of the video bit rates only once before the start of the distribution, additional overhead generated by exchanging throughput information between the sources is avoided. Therefore, the solution allows that the upload capacity of the sources is only used to distribute the videos. If the upload capacity of peers R_1 and R_2 changes, additional signals are of course required.

5.5.2. Evaluation

To verify the performance of the proposed scheme, an experimental prototype with two source peers and two requesting peers has been implemented. Two scenarios are evaluated:

- 1. Sources with independent rate allocation for different rate streams
- 2. Sources with joint rate allocation for same rate streams or alternatively same video quality streams

The first scenario shows the performance of the multi-source multicast scheme implemented in sources without coordination, while the second scenario shows the performance of the proposed multi-source multicast scheme using an adaptive mechanism to control the rate allocation between the sources.

Both scenarios are evaluated in terms of throughput, *PSNR*, and delay for all video streams. The adaptive multi-source multicast is based on a joint rate allocation decision and an adaptive rate mechanism that concurrently controls the bit rate for both sources. The independent rate allocation approach represents the case where the sources individually optimize their local upload capacity without coordination. The performance of the collaborative multi-sources multicast prototype has been evaluated in the PlanetLab infrastructure [153]. The first scenario introduced in Section 5.4, which corresponds to the joint rate allocation for different rate streams or different video quality streams is not implemented because this case represents an ideal situation where the upload capacity of every peer is exactly known and the download capacity of every node is infinite.

The videos sequences used to evaluate both approaches in the PlanetLab infrastructure are: Mother and Daughter, and Foreman. The short Foreman sequence is concatenated to a long test sequence with 3000 frames. The same is done with the Mother and Daughter sequence. Both video streams are encoded with the JSVM software [182] with the same video quality (PSNR) around 42 dB, but using different encoding rate and different number of layers for each sequence. To achieve this video quality the Foreman sequence needs a bit rate of 1600 kbps, which is obtained by using one base layer and two enhancement layers. The Mother and Daughter sequence is encoded at 230 kbps using one base layer and one enhancement layer. Both videos have the same duration (60 seconds), but the size of the Foreman and Mother and Daughter files are 10 MB and 1.5 MB, respectively. After this, a simple and small multicast group formed by four PlanetLab nodes is selected. The test multi-source framework is composed of two source peers (S_1 and S_2) and two requesting peers (R_1 and R_2). Helper peers are not considered in these experiments. The tested scheme is shown in Figure 5.11. The source S_1 is located at Massachusetts Institute of Technology (planetlab7.csail.mit.edu), while the source S_2 is located at University of Puerto Rico (node2.planetlab.uprr.pr). The requesting peers R_1 and R_2 are located at University of Oregon (planetlab2.cs.uregon.edu) and Stanford University (planet1.scs.stanford.edu), respectively. All the participating PlanetLab nodes are fully interconnected, including the sources nodes. In the first set of experiments, the adaptive multicast approach is evaluated in terms of delay for the two different videos. To this end, the two test videos with the same PSNR are allocated as video 1 and video 2 at the sources S_1 and S_2 , respectively. After this, the initial rates in the prototype multi-source multicast are fixed to 230 kbps and 1600 kbps for sources S_1 and S_2 , respectively. Mother and Daugther video is located at source S_1 , while Foreman video is located at source S_2 Then, both sources deliver their video files to all participating peers, including the sources.



Figure 5.11.: Arrangement of PlanetLab nodes during the evaluation of multi-source video multicast

Figure 5.12 shows the resulting throughput for both videos obtained from the experiments on PlanetLab. Figure 5.12 a) shows the video throughput obtained from the multi-source multicast with independent rate allocation, while the obtained throughput for both videos when using an adaptive mechanism for joint rate allocation introduced in section **??** is shown in Figure 5.12 b).

The results show that when adaptive rate mechanism is used, the difference between the delivery delays for both videos is greatly reduced in comparison to sources with independent rate allocation. In other words, the adaptive rate control allows us to integrate more collaboration between the sources in order to achieve a similar *PSNR* quality for all video sequences. The average throughput is used to recalculate the *PSNR* with JSVM software, obtaining *PSNR* values of 43.4 dB and 43.9 dB for Foreman and Mother and Daughter, respectively. Appendix A contains additional information about these experiments.

Figure 5.13 compares the throughput for the multi-source approach and the multi-source approach using independent rate allocation for the requesting peer located at University of Oregon. The results show that when the proposed approach is used, the requesting peer receives both videos in a more similar duration than when using independent sources.

Figure 5.14 a) shows how the sources with independent rate allocation tend to distribute their upload capacity equally between both sequences. Contrary, in Figure 5.14 b), the adaptive rate control greatly enforces the rate allocation in each source in order to assign more upload capacity to the video with higher bit rate (Foreman video), while the upload capacity allocation for the video with lower bit rate (Mother and Daughter) is reduced.



Figure 5.12.: Throughput comparison of the multi-source approaches delivering two different videos. *a*). Sources with independent rate allocation, *b*). Sources using an adaptive mechanism for joint rate allocation.



Figure 5.13.: Receiver throughput comparison for the requesting peer located at University of Oregon receiving two different videos. *a*). Independent rate allocation, *b*). Joint rate allocation using an adaptive mechanism.

In the second set of experiments, both sources distribute the same video (e.g. Foreman) and the adaptive control mechanism must enforce the same throughput for both video sequences in order to distribute both videos with the same quality (PSNR). Figure 5.15a) shows the performance of adaptive multi-source multicast when two identical videos sequences are multicast to all peers. The results show that a similar throughput of around 1.3 Mbps is achieved for both videos, therefore both videos are delivered with a similar PSNR to all peers. Contrary, the multi-source multicast with independent rate allocation does not always reach the same throughput for both sequences, because its performance is affected by the heterogeneous upload capacity of the sources or the dynamic behaviour of the network (see Figure 5.15b)).



Figure 5.14.: *Rate allocation percentage for each source: a). Sources with independent rate allocation, b). Sources using an adaptive mechanism for joint rate allocation.*

5.6. Chapter Summary

In this chapter, a novel approach for video streaming from multiple sources to multiple receivers in Peer-to-Peer (P2P) networks has been proposed and evaluated. The proposed scheme is suited for collaborative streaming environments, where the system inherently has multiple senders and video sequences and similar video quality is desired. In multi-source video multicast, all the participating peers are fully interconnected, and they are in fact receivers and senders at the same time. For the proposed approach, an analytical framework is developed for four different scenarios: sources with joint rate allocation for different rate or video quality streams, sources with joint rate allocation for same rate or video quality streams, sources with independent rate allocation for same rate streams, sources with independent rate allocation for different rate streams. In the analysis, the rate allocation problem is formulated as an optimization problem with an objective function that either maximizes the throughput or the aggregate video quality of the system.

The optimization problem is solved using linear programming for four different scenarios.



Figure 5.15.: Throughput comparison for the case of identical videos using adaptive multi-source multicast. *a*). Sources using an adaptive mechanism for joint rate allocation, *b*). Sources with independent rate allocation.

The results from extensive simulations show that multiple-source multicast with joint rate allocation achieves an optimal performance in terms of overall throughput. Furthermore, the *PSNR* enforcement helps us to balance the video quality of all video streams received by the participating peers. Its performance is similar to the maximum possible performance, and better than the performance achieved by the multi-source multicast with independent rate allocation.

Additionally, the chapter presents an adaptive mechanism, which compares the bit rate requirements for each delivered video in order to enforce the rate allocation in each source. The adaptive mechanism has been integrated in the multi-source multicast model to evaluate the performance of the sources with joint rate allocation on PlanetLab. To compare the performance of the adaptive mechanism, a multi-source model for sources with independent rate allocation also has been evaluated on PlanetLab. The results show that the adaptive multisource multicast provides a better rate adaptation in each source for different or same video sequences in comparison to sources with independent rate allocation.

5. Multi-Source Video Multicast

6. M-ary Tree-based Video Multicast

In this chapter an alternative solution based on multiple m-ary trees is proposed. The solution builds multiple m-ary trees in which the upload capacities of the peers are fully exploited and proximity issues are explicitly considered. In m-ary tree-based video multicast, the m-ary trees are rooted at the source and they always maintain a height of two levels from the root in order to avoid deep structures. The source splits the content into blocks and distributes every block separately by using a collection of m-ary trees. Every participating peer can receive one or more data blocks directly from the source. After this, every peer forwards the data block to its children in the corresponding m-ary tree. All peers contribute with their full upload capacity by being a forwarding peer in at least one of the m-ary trees. The goal of m-ary tree-based multicast is to maximize the overall throughput, while minimizing end-to-end delay by considering peer proximity issues during a multicast session.

The obtained m-ary tree collection is different to the tree collection used by Mutualcast because this scheme avoids that a single peer has to forward its received blocks from the source to all peers in the fully connected topology. Hence, the worst case delay encountered in Mutualcast does not apply to this scheme. The performance of the proposed scheme is determined using linear programming. Additionally, a prototype m-ary tree-based multicast has been evaluated in the PlanetLab infrastructure [153]. The performance of m-ary tree-based multicast has been compared with Mutualcast, which is the best known scheme in terms of maximizing overall throughput. The results show that the proposed scheme achieves an overall throughput similar to fully connected schemes such as Mutualcast, while maintaining a reduced end-to-end delay.

This chapter is organized in the following way:

• Motivation

The chapter starts by declaring the motivation and introduces an approach based on m-ary trees. In particular, proximity issues are discussed and how the Mutualcast approach can be affected when the participating peers are located in different geographical regions. Then, the concept of the m-ary tree based and proximity-aware approach is introduced.

• System Architecture

Here, the approach is presented and the algorithm to build the collection of m-ary trees for a multicast session is described.

• Analytical Model

A mathematical model for Application Layer Multicast based on multiple m-ary trees is developed which utilize the full upload capacities of all participating peers in order to maximize the overall throughput.

• Simulation

To simulate the performance of the proposed scheme, the distribution problem is formulated as an optimization problem and translated into a linear program, which can be solved using efficient optimization techniques.

• PlanetLab Implementation

The implementation of m-ary tree-based video multicast on PlanetLab is described, and extensive experimental evaluations of the proposed scheme are presented.

The author's research contributions in this chapter are:

- 1. A cooperative multicast model based on a collection of m-ary trees which achieves an overall throughput similar to fully connected schemes such as Mutualcast, while maintaining a reduced end-to-end delay.
- 2. An algorithm to obtain a near optimal m-ary tree collection rooted at the source which exploits the full upload capacity of all participating peers in combination with their proximity information during a multicast session.
- 3. A mathematical model for a proximity-aware collaborative Multicast and its formulation as an optimization problem to be solved using linear programming.
- 4. An implementation and real evaluation of an proximity-aware collaborative Multicast based on m-ary trees using the PlanetLab wide-are testbed, and its comparison with the Mutualcast performance.

6.1. Motivation

Peers can greatly benefit from the capacity of other requesting peers via collaboration, and hence the need to collaborate for multicast applications in large-scale and heterogeneous environments. Mutualcast is a fulling collaborative scheme, which engages as many peers as possible and uses their full upload capacities in order to maximize the overall throughput. However, the participating peers are typically in different geographical locations, such as is shown in Figure 6.1. Although Mutualcast achieves the maximum possible multicast throughput in P2P networks with constrained upload capacities, it does not address proximity issues.



Figure 6.1.: Peers are located in several geographical locations during a multicast session

M-ary Tree-based Video Multicast is inspired by Mutualcast, and it is extended by adding proximity information. In this work the Round-Trip-Time (RTT) between two peers is used as proximity information. In Mutualcast, the source assigns each block of content to a single peer for redelivery. Each peer redelivers its assigned block to the rest of the requesting peers. In this case, the distribution tree has two levels from the source for each data block. Thus, when all requesting peers have the same proximity among them, all blocks are delivered within the same time to all peers. The number of requesting peers in the multicast group is denoted as N_1 . This case is shown in Figure 6.2 where the source *S* distributes the blocks X_1 to X_4 to four requesting peers R_i . The distance *d* (the proximity measure) among them is assumed to be identical.



Figure 6.2.: Mutualcast with same distance d (e.g., RTT) among peers

Mutualcast does not control the delivery time. This is because each peer forwards its block to all other peers without considering that these peers may be distant from it. On the contrary, the proposed approach distributes a block through two or more peers, but they evaluate the peer proximity before forwarding their blocks to the rest of the peers. The basic idea is illustrated in Figure 6.3 using one source and five requesting peers R_i . The distance among peers is indicated by d, while X_i indicates the block management in each peer. Peers R_1 and R_2 have a distance of d from the source. Peers R_3 and R_4 have a distance of 2d from the source. Peer R_5 is 3d away. The worst case distance occurs when block X_5 from the source is delivered through peer R_5 to peer R_4 . In Figure 6.3 a), block X_5 travels a maximum distance of 8d when the Mutualcast approach is used. On the other hand, Figure 6.3 b) shows that when the proposed approach is used block X_5 encounters at most a distance of 5d during delivery from the source to all the requesting peers. How to build the collection of m-ary trees that achieves the previously described improved worst case distance is explained in the following section.

6.2. System Architecture

The design assumes that the source and all requesting peers collaborate by contributing their upload capacity. Asymmetric network access speeds (e.g. DSL) are also assumed and hence the upload capacity is considered to be the limiting resource. Each peer can contribute to the data distribution in one or more m-ary trees. Similar to Mutualcast, the source splits the content into blocks. After this, the system builds a collection of m-ary trees over which



Figure 6.3.: a). Illustration of the worst case distance for Mutualcast, b). *m*-ary tree-based approach when block X_5 from the source is delivered

the source delivers the content blocks. Every peer receives one or more data blocks from the source. Initially, peers with large upload capacity are used as forwarding peers, while peers with small upload capacity are placed as leaves in most multicast trees within the tree collection. When the remaining upload capacity of the best peers is small, they become leaves in the remaining trees.

The design assumes that the source knows the IP address and the upload capacity of all peers. Additionally, the distance among peers is assumed to be known. In this case, the distance measure is the round-trip time (RTT). For each participating peer R_j the upload capacity C_j is stored in list $C = C_1, \ldots, C_j, \ldots, C_{N_1}$ while the distance $D_i(j)$ between peers R_i and R_j is stored in list $D_i = D_i(1), \ldots, D_i(j), \ldots, D_i(N_1)$. For each peer R_j , every peer R_i calculates the normalized distance as

$$D_i^n(j) = \frac{D_i(j)}{\max_{D_i(j) \in D(i)} \{D_i(j)\}}$$
(6.1)

where $0 < D_i^n(j) \le 1$

The normalized upload capacity C_i^n for every peer R_j is computed as

$$C_j^n = \frac{D_j}{\max_{C_j \in C} \{C_j\}}$$
(6.2)

where $0 < C_i^n(j) \le 1$

The normalized distance values $D_i^n(j)$ and the normalized upload capacity values C_j^n are stored in lists $D_i^n = \{D_i^n(1), \ldots, D_i^n(j), \ldots, D_i^n(N_1) \text{ and } C^n = \{C_1^n, \ldots, C_i^n, \ldots, C_{N_1}^n\}$, respectively. These lists will be used by the tree-construction algorithm to select peers. Peers with high C_j^n and low $D_i^n(j)$ have a high preference to be selected. A preliminary delivery rate (PDR) is used to approximately determine how many times a peer can be used as a forwarding peer in different distribution trees. The PDR is computed as

$$PDR = \frac{\sum_{j=1}^{N_1} C_j}{N_1(N_1 - m)}$$
(6.3)

where $N_1 > m$.

 N_1 is the number of requesting peers in the multicast group and m is the number of peers directly connected to the root of an m-ary tree. The denominator of (6.3) represents the number of leaf peers being served by the forwarding peers in the set of distribution trees. In every tree there are $(N_1 - m)$ leaf nodes. Hence, the number of leaf peers in a set of m-ary trees, where the number of trees is the same as the number of requesting peers, is equal to $N_1(N_1 - m)$.

6.2.1. Building a Collection of m-ary Trees

The proposed approach is based on a heuristic construction of m-ary trees where all the participating peers collaborate with their upload capacity and their proximity information. Figure 6.3 depicts the algorithm to build the m-ary tree collection.



Figure 6.4.: Building a collection of *m*-ary trees.

Initially, the number of distribution trees to be obtained is identical to the number of requesting peers N_1 . To avoid deep structures the height h of the m-ary trees is fixed to two levels. Thus, the maximum number of requesting peers in each m-ary tree is m(m + 1). Assuming that the number of requesting peers in the multicast group N_1 is known, the source determines the degree m to be used as

$$m = \lceil \frac{1}{2}\sqrt{4N_1 + 1} - \frac{1}{2} \rceil$$
 (6.4)

After this, the source selects m children as the forwarding peers for each distribution tree T_i . The algorithm to select forwarding peers works as follows. The source calculates $D_i^n(j)$ and C_i^n in (6.1) and (6.2) for each peer and the preliminary delivery rate PDR in (6.3). The source selects the m peers with the largest normalized upload capacities C_i^n to be the forwarding peers in the first m-ary tree. The $(N_1 - m)$ leaf peers are assigned to the forwarding peers based on the proximity information. At most m leaf peers can be assigned to one forwarding peer. If the number of leaf peers is smaller than m^2 , then the assignment of leaf peers is additionally balanced to avoid exhausting one peer in one tree. For every leaf node, the source then subtracts one time the PDR from the upload capacity of the forwarding peers. Selected peers are used as forwarding peers in several distribution trees until their remaining upload capacity is no longer sufficient for building the next tree. The source then selects the next peer with the largest C_i^n from C^n which has not yet been used. When the next best peer has an upload capacity less than its number of children times *PDR*, the source calculates a new PDR which is obtained by dividing the peer's upload capacity between the number of leaf peers that it must feed. After this, the source uses this peer and exhausts the peer's upload capacity. Finally, when all peers have been used, but the m-ary tree collection is still not completed, the source reconsiders the peers that have still not been exhausted but this time using the most recent PDR value. In every step, when there are more than m peers with the same upload capacity, the source begins by selecting the *m* closest peers to it.

Out of the *m* closest peers, each peer distributes the received block to those who still have not received the same block from another forwarding peer in the distribution tree. The selection of these children is based on their proximity in order to avoid adding long delay. This helps us to reduce the end-to-end delay. The collection is completed when the number of obtained m-ary trees is equal to the number of peers N_1 . After this, identical m-ary trees are detected and deleted in order to obtain a reduced m-ary tree collection. A reduced mary tree collection allows us to reduce the number of blocks to be sent by the source and to increase their size. Once the reduced m-ary tree collection is obtained, linear programming (LP) can be used to compute the optimum block sizes that maximize the throughput Θ of the distribution tree collection. As explained later in this chapter, the set of multicast trees with their node upload capacities are translated to decision variables and constraints of a linear program. The delivery latency is the end-to-end delay from the source to the receivers. The delay minimization has already been taken care of during tree construction.

6.2.2. Example

This section illustrates the algorithm described in the last section with an example. The example assumes a multicast group with seven requesting peers R_i and a sender S. The upload capacity C_i of the requesting peers R_i and their distance from the source is given in Table 6.1.

R_i	C_i	$D_{source}(j)$	C_j^n	$D_{source}^n(j)$
	(kbps)	(ms)	Ŭ	
1	100	1495	0.2	0.633
2	500	2361	1.0	1.0
3	300	1716	0.6	0.726
4	400	1731	0.8	0.733
5	200	913	0.4	0.386
6	300	1200	0.6	0.508
7	200	390	0.4	0.165

Table 6.1.: Upload capacity C_i of the requesting peers and their proximity to the source. the two right main columns show the normalized values (compare (6.1) and (6.2))

The distances (RTT) among the requesting peers and to the source are obtained from the network coordinates model proposed in [212] with data acquired from CAIDA's Skitter project [213]. Initially, based on the number of requesting peers, seven m-ary trees are created to distribute the seven blocks X_1 to X_7 . Using (6.1) and (6.2), each peer and the source normalize the upload capacity and distance for every participating peer. Table 6.1 shows the normalized upload capacities and distances for the source only, but a similar operation is realized for each peer. Thus, the details for the rest of the peers is skipped. The *PDR* and *m* values for the source are calculated from (6.3) and (6.4) to be 71.4 kbps and 3, respectively. Since the intention is to build m-ary trees which are balanced as much as possible, then in the example, distribution trees with one forwarding peer feeding two leaf peers and two forwarding peers feeding one leaf peer each are obtained. In Figure 6.5, R_2 , R_4 and R_6 are used as forwarding peers in the first m-ary tree to distribute block X_1 .



Figure 6.5.: Preliminary m-ary tree collection

This is because peers R_2 , R_4 and R_6 have the largest upload capacities in the multicast group. Although peers R_6 and R_3 have the same upload capacity, R_6 is preferred because it

has a smaller normalized distance to the source than peer R_3 . The rest of the nodes receive block X_1 from the closest forwarding peer. Thus, peer R_2 selects peers R_3 and R_5 as its two closest leaf peers to forward block X_1 , while R_6 and R_4 sends block X_1 to peers R_7 and R_1 , respectively. The remaining upload capacity of R_2 , R_4 , and R_6 , becomes 357.2 kbps, 328.6 kbps and 218.6 kbps, respectively. Comparing the remaining peer upload capacity to the *PDR*, the source determines that peers R_2 , R_4 and R_6 can still be used as forwarding peers in three additional distribution trees. In the fourth distribution tree, the capacity of R_2 is exhausted, while the available capacity of R_4 and R_6 is reduced to 43 kbps and 14.4 kbps respectively, which is smaller than *PDR*. Therefore, R_4 and R_6 cannot be used in another distribution tree for the time being. Now, the source determines that peers R_3 , R_5 and R_7 can be used as the next forwarding peers in the fifth and sixth distribution trees. Afterwards, the remaining upload capacity of peers R_3 , R_5 and R_7 becomes 14.4 kbps, 57.2 kbps and 57.2 kbps, respectively. Because the remaining upload capacity in these peers is smaller than PDR, they cannot be used in another distribution tree for now. The source then selects the next peer not yet used, which is peer R_1 . However, this peer has a capacity less than 2 * PDR and thus in order to use this peer as forwarding peer, the source must adjust the *PDR* to half the capacity of the peer. After this, the source exhausts the capacity of the peer. Once all peers have been used as forwarding peers in at least one distribution tree, but the number of obtained m-ary trees is still different from the number of nodes, the remaining upload capacity of peers R_5 and R_7 need to be exhausted using the new PDR. The source then selects R_1 , R_5 and R_7 as forwarding peers in this last distribution tree. Each forwarding peer sends their received block(s) to their closest leaf peers. The obtained m-ary tree collection is shown in Figure 6.5. Figure 6.5 shows how the source uses the same m-ary tree structure for the delivery of blocks X_1 , X_2 and X_3 . The same situation holds for blocks X_5 and X_6 . Here, duplicate trees can be eliminated in order to obtain a reduced m-ary tree collection, and the source now can distribute its content using four m-ary trees only. Each distribution tree delivers a specific block to all requesting peers. The resulting reduced tree collection is shown in Figure 6.6.



Figure 6.6.: Reduced m-ary trees collection

The maximum throughput of the tree collection and the size of the blocks are determined

using linear programming. Figure 6.7 shows the reduced m-ary tree collection from Figure 6.6 translated into a linear program.

Maximize

$$\Theta = X_1 + X_2 + X_3 + X_4 + X_5$$

subject to
c1: $3X_1 + 3X_2 + 3X_3 + 3X_4 + 7X_5 <= 2000$
c2: $2X_4 <= 100.0$
c3: $2X_1 + X_2 <= 500.0$
c4: $2X_3 <= 300.0$
c5: $X_1 + 2X_2 <= 400.0$
c6: $X_3 + X_4 <= 200.0$
c7: $X_1 + X_2 <= 300$
c8: $X_3 + X_4 <= 200$

Figure 6.7.: A reduced *m*-ary tree collection is translated into a linear program for throughput maximization

The source splits the content into five blocks. The blocks X_1 to X_4 are distributed from the source to the requesting peers through four m-ary trees, while the block X_5 is distributed from the source to each requesting peer directly. This example assumes that the source has an upload capacity of 2000 kbps. The first constraint $3X_1+3X_2+3X_3+3X_4+7X_5 \leq 2000$ kbps considers the upload capacity of the source, which has to deliver three blocks to every m-ary tree and additionally sends the block X_5 to every peer directly. The rest of the constraints consider the upload capacity of the requesting peers R_1 , R_2 , R_3 , R_4 , R_5 , R_6 and R_7 . The solution gives a maximum throughput of 571.4 kbps, while the size of the blocks in kbits is $X_1 = 200$, $X_2 = 150$, $X_3 = 50$, $X_4 = 100$ and $X_5 = 71.42$, respectively.

For this specific example, the solution reaches the same maximum throughput as Mutualcast. However, this is not always the case since a heuristic approach is used to find a near-optimal solution. Calculating the optimal set of multicast trees on the fly is a hard task, because an exact solution requires to evaluate all possible combinations for all co-existing sessions in the overlay network and the number of combinations and constraints grows exponentially with the number of participating peers.

Because the collection of m-ary trees is based on heuristics an exact solution is not always possible. However, the study found through extensive simulations that in cases when balanced m-ary trees are used and the source capacity is abundant the proposed approach can achieve an overall throughput identical to the maximum throughput given by Mutualcast.

6.3. Simulation Results

The approach is evaluated in terms of overall throughput and delivery latency. The maximum overall throughput is computed using linear programming as explained in the previous section. The maximum required time so that all the nodes receive all data blocks is obtained too. The results obtained with the proposed approach are compared with the results when using the Mutualcast approach. Different cases are simulated for multicast groups with 6, 7 and 10 requesting peers. In every case, the source capacity is varied from 1000 to 6000 kbps, while the upload capacity of each requesting peer is fixed. Heterogeneous upload capacities and proximity relationships are used for every peer in the different cases. The upload capacity of the requesting peers R_1 , R_2 , R_3 , R_4 , R_5 , R_6 , R_7 , R_8 , R_9 and R_{10} in kbps is 100, 500, 300, 400, 200, 300, 400, 300 and 100, respectively. The round trip times between peers and the source are obtained from [212] and [213]. Helper peers H_i are not considered in this example. A comparison between m-ary trees rooted at the source and Mutualcast in terms of overall throughput for theses cases is shown in Figure 6.8.



Figure 6.8.: Overall throughput comparison

The results show that for a heterogeneous multicast group, the m-ary trees based approach achieves almost the same overall throughput as the Mutualcast approach. Both approaches consider that the source S has enough capacity and it can distribute extra content blocks directly to all requesting peers R_i . Both approaches lead to identical throughput if the source capacity has enough upload capacity. The proposed approach and the Mutualcast approach are compared in terms of delivery latency. Delivery latencies achieved by the proposed approach and Mutualcast are compared in Figures 6.9. Figure 6.9a) shows the maximum delivery delay by the proposed approach and the Mutualcast approach for the delivery of all content blocks to all requesting peers in the multicast group. This delay is determined by the slowest distribution tree. The results show that when the m-ary trees based approach is used, the maximum end-to-end delay is smaller in comparison to Mutualcast. This improvement is attributed to the fact that the algorithm based on m-ary trees avoids deep structures and incorporates proximity information into the overlay topology. Figure 6.9b) shows the average delivery time for all blocks for all peers. This delay is the sum of the maximum end-to-end delay in each distribution tree divided by the number of distribution trees. The approach based on m-ary trees rooted at the source shows a better average end-to-end delay than the Mutualcast approach.

Finally, the delivery rate achieved by each distribution tree in the proposed scheme and the Mutualcast scheme is evaluated during a multicast session. For this comparison, a set of six peers and a source capacity of 1000 kbps is considered. Here, both evaluated approaches achieve the same overall throughput. The evaluation assumes that each block X_i is delivered by a distribution tree. The results are shown in Figure 6.10. For both approaches, the blocks X_1 to X_5 within the square represent the blocks delivered by each distribution tree, while the



Figure 6.9.: Delay comparison between Mutualcast and multicast based on m-ary trees. a). Maximum delay b). Average delay

block X_7 is delivered by the source to each requesting peer directly. While in the proposed approach the source splits the total content in five blocks, Mutualcast requires to use an extra block X_6 . This is because in Mutualcast each block X_i is assigned to one single node for redelivery. The results show that both models use different delivery rates in each distribution tree to maximize the overall throughput. Most distribution trees in the proposed approach allow a bigger delivery rate than Mutualcast during a multicast session. This is because the proposed approach uses fewer blocks but with a bigger size by distributing via five distribution m-ary trees only, while Mutualcast uses six distribution trees.



Figure 6.10.: Delivery rate on each distribution tree

6.4. Planet Implementation and Evaluation

The performance of a prototype proximity-aware collaborative multicast based on m-ary trees has been evaluated in the PlanetLab infrastructure. The implementation runs over Linux and consists of different programs written in C/C++ language. In the PlanetLab experiments, the m-ary tree-based multicast scheme is compared to the Mutualcast scheme

and the performance is evaluated in terms of delivery delay. The evaluation assumes that the scheme with the smallest delivery time presents the best overall performance. Similar to [49], the implementation is based on the TCP protocol and uses a reduced number of requesting peers in the multicast group. The following cases are evaluated:

- 1. Throughput among all participating (source and requesting) peers is similar.
- 2. Throughput between the source and some subset of peers is highly heterogeneous.
- 3. Throughput among subsets of peers is weak.

In all these cases, a scenario with 6 requesting peers is considered. The selected sites to evaluate the different cases are shown in Figure 6.11. Fixed tree structures are considered to evaluate the m-ary tree-based multicast scheme and the Mutualcast scheme. The evaluation uses fixed structures for both schemes.



Figure 6.11.: PlanetLab nodes used in the USA and Europe to evaluate the proximity-aware collaborative multicast scheme and the Mutualcast scheme

In the experiments, the source distributes 60 frames of an MPEG2 video, which are obtained from a video file by using the SID video cutter and splitter software [214]. The Mutualcast scheme uses 6 blocks to distribute 60 frames to the requesting peers by allocating 10 frames in each block, while m-ary tree-based multicast uses 3 blocks to distribute 60 frames to the requesting peers by allocating 20 frames in each block. The average size of the blocks in Mutualcast is 67 KB, while in the proposed scheme the average size of the blocks is 130 KB. The experiment assumes that the blocks are small media files and then they are broadcasted via the m-ary tree-based multicast approach or the Mutualcast approach from the source to all requesting peers. Here, the measurements are based on the required time to distribute these blocks to all peers unlike to the results presented in the previous section and in [123] which are based on the Round-Trip Time (RTT) only.

During the implementation on PlanetLab, the program runs the following protocol: First, each requesting peer requests the same video from the source. The source copies the IP address of every requesting peer and sends all IP addresses to all requesting peers. Using this information, the source and each requesting peer obtains the RTT by performing a PING among all participating peers. The source sends a video block to every requesting peer. After the blocks are received at each requesting peer, each requesting peers forwards the blocks to its leaf peers in a parallel way. The Mutualcast implementation on PlanetLab is similar, but each requesting peers in the multicast group. In both approaches, the distribution of blocks among the requesting peers is implemented using threads. In parallel, each peer runs a thread for each requesting peer to which it must forward the received blocks.

For all evaluated cases, exhaustive measurements have been realized and average values are considered. The cases compare the maximum and average delays between the m-ary tree-based multicast approach and the Mutualcast approach.

6.4.1. Case 1: Throughput among All Participating Peers is Similar

The first case evaluates the multicast approach based on m-ary trees and the Mutualcast approach in terms of delivery time when the throughput between the source and all peers is similar. The source is located at Rice University (ricepl-1.cs.rice.edu) while the set of requesting peers is formed by PlanetLab nodes located at the University of California, Santa Barbara-UCSB (planet1.cs.ucsb.edu), Brown University (mercury.cs.brown.edu), University of California, San Diego-UCSD (planetlab3.ucsd.edu), Hewlett-Packard Labs (HP-PA) in Palo Alto (pli1-pa-3.hpl.hp.com) and Boston University (planetlab-01.bu.edu (BOSTON1) and planetlab-02.bu.edu (BOSTON2)). A sub-set of requesting peers is formed in California, while another subset is formed on the east coast of the United States (Brown University and Boston University). Boston University has two hosts as requesting peers, while the rest of the sites have a host as requesting peer only. Using the algorithm detailed in Section 6.2.1, the blocks X_1 to X_3 are distributed through three distribution m-ary trees as is shown in Figure 6.12.



Figure 6.12.: m-ary tree collection when all participating peers present a similar throughput

Using this m-ary tree collection and the Mutualcast scheme, extensive measurements on PlanetLab have been realized. Figure 6.13 compares the maximum delay between the proposed approach and the Mutualcast approach to distribute all content blocks to all requesting peers. The maximum delay is determined by the slowest distribution tree, while the average delay is the sum of the maximum end-to-end delay in each requesting peer divided by the number of requesting peers.



Figure 6.13.: Delivery delay to distribute all blocks when the throughput among all peers is similar. The source is located at Rice University

The results show that in most of the measurements, Mutualcast presents better distribution times than the scheme based on m-ary trees. The maximum delays in Mutualcast are reduced between 5% and 15% compared to the scheme based on m-ary trees. This is because in this scheme the size of the blocks sent by the source is twice as large as in Mutualcast, thus creating a bottleneck at this point. Hence, in spite of the delivery times among peers in m-ary tree-based multicast being smaller than in Mutualcast, the final distributed times are larger. For this case, the Mutualcast solution is better than the scheme based on m-ary trees.

6.4.2. Case 2: Delivery Delay when the Throughput between the Source and a Subsets of Peers is Highly Heterogeneous

The second case evaluates the delivery time when the throughput between the source and a subset of requesting peers is highly heterogeneous. To this end, an m-ary tree collection similar to the previous case is considered, but the source is located in the geographical region of a subset. Just like the previous case, there are two subsets of requesting peers, each in a specific geographical region such as California and the east coast of the United States. In this experiment, the source is a PlanetLab node hosted at the Massachusetts Institute of Technology - MIT (planetlab7.csail.mit.edu) located at the east coast. Therefore, the source is close to PlanetLab nodes at Brown University (mercury.cs.brown.edu (BROWN)) and Boston University (planetlab-01.bu.edu (BOSTON1) and planetlab-02.bu.edu (BOSTON2)). The resulting arrangement is shown in Figure 6.14. This arrangement has been used to make extensive measurements on PlanetLab.

Figure 6.15 compares the delivery delay by m-ary tree-based multicast and Mutualcast for delivery of all content blocks to all requesting peers in the multicast group.

The results show that when the proposed approach is used, the maximum delivery time is smaller in comparison to Mutualcast. Using m-ary tree based multicast the maximum delivery time is reduced by 30% with respect to Mutualcast in sample 3. This improvement is attributed first to the fact that the source is close to a subset of requesting peers on the Plan-



Figure 6.14.: m-ary tree collection when the throughput between the source and a subset of peers is highly heterogeneous



Figure 6.15.: Delivery delay to distribute all blocks to all peers when the throughput between the source and a subset of peers is highly heterogeneous. The source is located at the MIT

etLab infrastructure. Thus, the throughput between the source and a subset of requesting peers is larger than the throughput among the requesting peers. The second fact is that the algorithm based on m-ary trees incorporates proximity information into the overlay topology avoiding the communication between distant peers. When this case is present in a P2P based multicast session, the approach based on m-ary trees performs better than the Mutualcast approach in terms of delivery times.

6.4.3. Case 3: Delivery Delay when the Throughput between the Subsets of Peers is Weak

The third case evaluates the delivery time when the throughput between the subsets of requesting peers is weak. To evaluate the third case, the experiments use PlanetLab nodes hosted in the USA and Central Europe. A subset of requesting peers are hosted on different PlanetLab nodes located in the California region, specifically at UC Santa Barbara (planet1.cs.ucsb.edu (UCSB)), UC San Diego (planetlab3.ucsd.edu (UCSD)) and HP Labs in Palo Alto (planetlab3.ucsd.edu (HP-PA)), while the second subset of requesting peers located in Central Europe is deployed on PlanetLab nodes hosted at INRIA Labs in Sophia-Antipolis (planetlab1.inria.fr (INRIA1) and planetlab1.inria.fr (INRIA2)), Czech Education and Research Network (planetlab1.cesnet.cz (CESNET)) and University of Copenhagen (planetlab2.diku.dk (DIKU1)). In this case two sets of experiments are realized. In the first experiment, PlanetLab nodes located in different sites are used, while in the second experiment two used PlanetLab nodes are located in the same site (e.g. planetlab1.inria.fr (INRIA1) and planetlab1.inria.fr (INRIA2)). In the first experiment, the m-ary tree collection used to run both approaches is shown in Figure 6.16. Three m-ary trees are used to distribute the blocks X_1 to X_3 from the source to each requesting peer.



Figure 6.16.: m-ary tree collection to evaluate the first experiment when throughput among subsets of peers is weak. The source is located at the MIT

Figure 6.17 compares the maximum delays between the proposed approach and Mutualcast with data obtained from extensive evaluations.



Figure 6.17.: Delivery delays obtained from the first experiment when throughput among subsets of peers is weak. Source is located at the MIT

In the first experiment, the results show that when there are six requesting peers deployed in the USA and Europe and balanced m-ary trees (3 peers in the USA and 3 peers in Europe) in the multicast group are used, the proposed scheme reaches smaller delivery times than the Mutualcast scheme in most of the peers. For example, using m-ary tree-based multicast the maximum delay is reduced by 6% with respect to Mutualcast. This is because the links to the INRIA PlanetLab node (planetlab1.inria.fr) are working very slowly. Additionally, mary tree-based multicast reaches a better average delay than Mutualcast in all samples. This means that most of the requesting peers obtain the content blocks more quickly using the scheme based on m-ary trees. Also, Figure 6.17 shows that the proposed approach reduces the delivery times by about 25% compared to Mutualcast for the peers deployed in California. This improvement is attributed to the fact that the algorithm divides the multicast group in two subsets and the requesting peers in California are not affected by delay introduced from some slow peers in Europe.

For the second experiment, a subset of three requesting peers is deployed on different PlanetLab nodes hosted in California as in the previous experiments. The rest of the requesting peers are deployed in Central Europe, specifically two peers run in PlanetLab nodes (planetlab1.inria.fr and planetlab2.inria.fr) hosted in INRIA while one peer runs on a CESNET's PlanetLab node (planetlab1.cesnet.cz). The m-ary tree collection used to distribute the blocks X_1 to X_3 from the source to each requesting peer is shown in Figure 6.18.



Figure 6.18.: m-ary tree collection to evaluate the second experiment when throughput among subsets of peers is weak. The source is located at the MIT

The maximum delays between the proposed scheme and the Mutualcast are compared in Figure 6.19. Similar to the first experiment in this case, the results in the second experiment show that m-ary tree based multicast presents better end-to-end delays than Mutualcast. However, in the second experiment the difference between the delivery times in both schemes is larger. The results show that when using m-ary tree-based multicast the maximum delay is reduced by 27% with respect to Mutualcast. In all samples, the proposed approach reaches a better average delay than the Mutualcast approach. Also, using the proposed approach most of the peers in both regions obtain all content blocks with smaller delivery time than using the Mutualcast approach. This improvement is attributed to the fact that m-ary tree-based multicast avoids to use weak links among the requesting peers, using the strong links among peers only. For this case, the approach based on m-ary trees leads to lower delivery times than Mutualcast.

6.5. Chapter Summary

Finding a good tree topology that maximizes the overall throughput and limits delivery delay is critical in delay sensitive multicast applications. In this chapter, a content distribution approach based on m-ary trees has been proposed and evaluated.

The heuristic tree construction approach generates a collection of m-ary trees rooted at the source by combining the full upload capacities of all participating nodes and their proximity relationship based on round trip time (RTT). In the proposed model, all the requesting peers collaborate as relay peers in at least one tree, while their full upload capacity is exhausted.



Figure 6.19.: Delivery delays obtained from the second experiment when throughput among subsets of peers is weak. Source is located at the MIT

M-ary based video multicast aims to maximize the overall throughput while the maximum delay is reduced. To this end, the resulting m-ary tree collection is formulated as an optimization problem with an objective function that maximizes the throughput. Because the proposed solution is based on heuristic techniques, an exact solution is not always possible and an analytical framework is not required. The optimization problem is solved using linear programming

The m-ary tree based video multicast scheme is compared with the Mutualcast scheme which achieves the maximum possible throughput, but potentially leads to large delivery delays. For both approaches, extensive simulations were realized. The results show that for a heterogeneous multicast group, the proposed m-ary trees based approach achieves almost the same overall throughput as the Mutualcast approach, but a smaller delivery time.

Both approaches have been evaluated on PlanetLab using PlanetLab nodes deployed on different parts of the globe for different cases. Due to the dynamic behavior of PlanetLab, an exact measurement of the peers upload capacity is not possible. For this reason, the implementation uses the throughput among the participating peers, which results from sending a flow of a certain size divided by its duration, as an estimate for the upload capacity. Using the initial throughput, the source can send blocks of different size to each requesting peer. The results demonstrate that the multicast scheme based on m-ary trees presents a smaller end-to-end delay than Mutualcast. In addition, the proposed scheme avoids a fully-connected topology, while maintaining a scalable structure.

7. Conclusions and Future Work

7.1. Conclusions

Currently, many video delivery technologies such as unicast, broadcast and multicast exist. Broadcast video is an ideal technology to distribute a limited number of video streams to a very large audience. In contrast, multicast allows that the receiver get the desired content, while a significant large number of streams can be available [198]. Multicast also is motivated by the need to support one-to-many and many-to-many applications. However, several challenges such as routers migration and restricted connectivity have limited its widespread deployment in today's Internet [197]. Recently, P2P technology has emerged as a valuable infrastructure for video delivery to a very large audience.

This dissertation addresses issues of video multicast over P2P networks. In this dissertation, P2P networks have been selected as the distribution infrastructure because these networks do not require a special network infrastructure, are easy to deploy, and participants share their upload capacity for video delivery to other participants. Unfortunately, a number of technical challenges must be solved before P2P video multicast becomes viable [215], [164], [216]. In Chapter 2 and Chapter 3, conceptual issues related to media delivery and the state of the art in overlay-based distribution were discussed.

This dissertation addresses different challenges of video streaming over P2P networks such as low delay and video quality. The aim of this research work was to develop architectures for media streaming delivery. In the proposed solutions, multicast is addressed for oneto-many and many-to-many applications. Different parts of a video streaming architecture were investigated, but mainly, the work focuses on encoding and transport issues. The encoding rate has a relevant importance in video streaming systems with limited throughput, because a suitable encoding rate allows us to achieve sufficient video quality in these systems. These benefits are evaluated for multi-source multicast systems. The media streaming architectures proposed in this dissertation are inspired by Mutualcast [49], which is an efficient mechanism for one-to-many content distribution that maximizes the overall throughput by exploiting the upload capacity of all participating peers. The Mutualcast approach is extended to a hierarchical approach and a multi-source approach. Similar to Mutualcast, the proposed schemes are tailored to the small-scale ALM approaches.

A central contribution of this thesis is muti-source video multicast presented in Chapter 5. Multi-source multicast is a novel framework for video streaming from multiple sources to multiple receivers in Peer-to-Peer (P2P) networks. In multi-source video multicast, the goal is to maximize the overall throughput or alternatively the aggregate video quality of multiple concurrent streaming sessions. To this end, an analytical model for the novel multi-source video multicast scheme has been presented and it is backed up by numerical experiments. The analytical model studies the multi-source multicast system in terms of throughput and *PSNR* for four different scenarios. The rate allocation problem in the multi-source multicast model has been formulated as an optimization problem with an objective function

to maximize the overall throughput or alternatively the aggregate video quality. The analytical evaluation shows that sources with joint rate allocation present a superior performance than sources with independent rate allocation. Results obtained from extensive simulations based on linear programming validate the analytical framework and show the effectiveness of the proposed multi-source video multicast scheme. Furthermore, the PSNR enforcement helps us to balance the video quality of all video streams received by the participating peers. To enforce the same *PSNR* for different videos, the sources must adaptively control their links. To this end, an adaptive mechanism has been integrated to the multi-source video multicast system. The adaptive mechanism compares the bit rate requirements for each delivered video in order to enforce the rate allocation in each source. Scalable video coding is used to adjust the same video quality among different video sequences from heterogeneous sources. However, the same *PSNR* for all videos can be achieved when the sources have enough upload capacity. The proposed scheme has been evaluated and compared with a multi-source scheme with independent rate allocation on PlanetLab. The results show that an adaptive framework provides a better rate adaptation in each source for different or same video sequences in comparison to sources with independent rate allocation and different rate streams. The proposed multi-source video multicast scheme is suited for collaborative streaming environments, where the system inherently has multiple senders and video sequences and similar video quality is desired.

Low delay is very important for delay sensitives applications. A video streaming system reaches low delivery time and high scalability by using hierarchical distribution structures as it is shown in Chapter 4. To achieve very low latencies, hierarchical collaborative multicast organizes all peers into small hierarchical clusters through a distribution tree. Hierarchical collaborative multicast has been evaluated and compared with Mutualcast on PlanetLab. Compared with Mutualcast, hierarchical collaborative multicast provides a lower delivery time and reaches a better scalability maintaining a reduced number of connections.

Mutualcast, Multi-source video multicast and hierarchical collaborative multicast are solution for media delivery based on fully meshed overlay networks. Another alternative for media delivery are forest-based approaches. In general, a media streaming system based on trees has reduced delivery time. However, finding a good tree topology that maximizes the overall throughput and limits delivery delay is critical in delay sensitive multicast applications. To this end, a content distribution approach based on m-ary trees in Chapter 6 is proposed and evaluated . The proposed approach is compared with the Mutualcast scheme which achieves the maximum possible throughput, but potentially leads to large delivery delays. A heuristic tree construction approach generates a collection of m-ary trees rooted at the source by combining the full upload capacities of all participating nodes and their proximity relationship. The results obtained from PlanetLab demonstrate that the proposed multicast scheme provides a good balance between reduced end-to-end delay and maximum overall throughput while maintaining a scalable structure by avoiding a fullyconnected topology.

All video streaming architectures proposed in this dissertation have been implemented and evaluated in a real scenario. To this end, the local network of the Institute of Communication Networks (LKN) and the PlanetLab infrastructure have been used. PlanetLab allows us to

evaluate prototypes in a wide-area network.

Some computational problems are hard, and some of them were faced during the research phase. For example, finding the optimal combination that maximizes the overall throughput of multiple trees [49], [206] or choosing the optimal clustering assignment are NP-hard problems [195], [206]. To deal with these NP-hard problems different solutions based on heuristic algorithms were proposed.

7.2. Limitations

Although the proposed schemes introduce novel ideas for media delivery, they mainly have the following limitations:

First, similar to Mutalcast, all the multicast models proposed in this dissertation are based on fixed topology, which is defined before to begining of the content delivery. A fixed topology policy is usually not affordable within P2P networks, because new peers cannot be incorporate to the multicast group during a multicast session.

Second, the links between peers in the multicast schemes were established via TCP as in the Mutualcast approach. In special for the distribution schemes studied in this dissertation, TCP offers several advantages as send and receive buffers, flow control, and reliable data delivery and control congestion. However, TCP mechanisms may introduce long delivery times and widely varying throughput, which are not suitable for real-time streaming applications.

7.3. Future Work

The popularity of media streaming over P2P networks is growing during last years, and these systems will continue being an interesting research topic in the near future. Several open issues are still to be addressed in these systems. Furthermore, a large number of multimedia applications require support for video multicast. The work presented in this dissertation can be extended in many directions:

Dynamic environment

In general, the fixed topology of the media streaming architectures proposed in this dissertation are not prepared for very dynamic environments, where the requesting peers dynamically join or leave the multicast group.

For dynamic environments, the hierarchical collaborative multicast system could integrate dynamic clustering into its hierarchical structure, where the requesting peers or helper peers join or leave the multicast group. Furthermore, the clusters formation process is a NP-hard problem [195], [206], and new and more efficient clustering algorithms are always desirable.

Media streaming architectures based on UDP

During the development of this dissertation, some experiments with a Mutualcast scheme based on UDP (User Datagram Protocol) have been realized. In these experiments, Mutualcast based on UDP achieves a better throughput than Mutualcast based on TCP. A flow control mechanism was implemented in order to replace the TCP flow control mechanism.

For the future work, an extension of the media streaming architectures proposed in this dissertation to UDP is recommended. This argument is based on the fact that TCP introduce delays that are unacceptable for real-time applications with stringent delay requirements. In media streaming architectures based on UDP, reliability issues must be reached adding redundant information in the original message stream by using some FEC (Forward Error Correction) technique.

Scalable video coding

In scalable video coding the source encodes the data stream into a base layer and several enhancement layers. The layers can be combined at the receiver to provide progressive refinement. In multi-source video multicast, scalable video coding (H.264/SVC) has been used to adapt the same video quality for different videos distributed from different sources to multiple requesting peers. Results show benefits by using scalable video coding. For the future work, the incorporation of scalable video coding techniques to the hierarchical collaborative and m-ary tree-based models presented in this dissertation is recommended.

In scalable video, each layer has different priority and its own minimum bit rate. Therefore, in order to achieve various video quality, upload capacity can be allocated to transmit different layers. Thus, peers with different upload capacity can receive different layers. For example, in hierarchical collaborative multicast, peers allocated into top clusters can receive a larger number of layers than peers allocated into inferior clusters. As the base layer plays an important role in the video quality, the system should make sure that the base layer is delivered to all requesting peers in the multicast group. For the enhancement layers, the system would consider that they could strongly depend of the quality level required by each peer.

An alternative solution to be implemented in the proposed architectures in this work is Multiple description coding (MDC).

The evaluation of all new media streaming architectures on PlanetLab is recommended, because PlanetLab allows us to evaluate the performance in a wide-are network and under real-world conditions.

Network coding

Network coding was proposed by Ahlswede et al. in [207]. During this research period, the benefits of network coding in the multicast systems were investigated. Using network coding, the broadcast capacity between a source and a set of requesting peers can always be achieved, but the intermediate peers must code their input packets to produce output packets [49].

Although a considerable theoretical work about network coding has been realized during last years, a limited number of practical systems (e.g. Avalanche [208], Practical network coding [209]) has been implemented. Currently, the scientific community discuss about the benefits of network coding in P2P networks [210] and several issues are still open.

For example, in network coding, data must be encoded at intermediate node, which means that total encoding overhead will be constantly increasing as the data travels through the network. Speed at which the data can be encoded and decoded is very important during content distribution, but it has been not considered in previous network coding research. On other hand, network coding for multi-source multicast networks [222] represents more interesting challenges than that for single source multicast networks. Future work can be realized in this direction.

Security and content protection

Some media streaming delivery systems require to protect the contents to be distributed over the Internet. However, most P2P networks do not provide Digital Right Management (DRM), which allows us to illegally share copyright contents from these systems. For the future work, integrating Digital Rights Management with the P2P-based media streaming architectures proposed in this dissertation is recommended. How the content to be shared in the P2P network can be identified is a key issue in a content distribution system. Although DRM and P2P networks are technologies of the Internet era, an integration between both technologies has received a limited attention [211]. Incorporation of DRM mechanisms into P2P networks represents an valuable opportunity to provide reliable content protection in the media streaming systems.

7. Conclusions and Future Work

A. Appendix

A.1. Simulation for Multi-source video Multicast

This section presents detailed results from the simulations for the multi-source video multicast presented in Chapter 5. Results presented in this appendix are based on throughput and video quality (PSNR).

For all cases, a set of 5 participating peers is used, where two peers S_1 and S_2 act as sources and three peers R_1 , R_2 and R_3 are the requesting peers. The upload capacity of the participating peers C_{S_2} , C_{R_1} , C_{R_2} and C_{R_3} in kbps is 600, 500, 300, and 300, respectively. The upload capacity C_{S_1} of the source S_1 is varied from 200 to 900 kbps. Helper peers are not considered in this example. The tested video sequences are: Foreman and Mother and Daughter.

A.1.1. Results for Throughput-based Simulation

Scenarios for simulation based on throughput are:

- 1. Sources with joint rate allocation for different rate streams,
- 2. Sources with joint rate allocation for same rate streams,
- 3. Sources with independent rate allocation for same rate streams,
- 4. Sources with independent rate allocation for different rate streams.

A.1.1.1. Sources with Joint Rate Allocation for Different Rate Streams

The results obtained from the simulations for sources with joint rate allocation and different rate streams are summarized in Table A.1.

C_{S_1}	C_{S_2}	Θ	X	Y
(kbps)	(kbps)	(kbps)	(kbps)	(kbps)
200	600	475	179.15	291.81
300	600	500	216.65	283.32
400	600	525	233.32	291.66
500	600	550	279.22	270.78
600	600	575	278.83	296.17
700	600	600	289.24	310.76
800	600	625	299.67	325.33
900	600	650	310.06	339.94

Table A.1.: Overall throughput Θ for sources with joint rate allocation and different rate streams

A.1.1.2. Sources with Joint Rate Allocation for Same Rate Streams

The results obtained from the simulations for sources with joint rate allocation and same rate streams are summarized in Table A.2.

Table A.2.: Overall throughput Θ for sources with joint rate allocation and same rate streams

C_{S_1}	C_{S_2}	Θ	X	Y
(kbps)	(kbps)	(kbps)	(kbps)	(kbps)
200	600	400	200	200
300	600	500	250	250
400	600	525	262.5	262.5
500	600	550	275	275
600	600	575	287.5	287.5
700	600	600	300	300
800	600	625	312.5	312.5
900	600	650	325	325

A.1.1.3. Sources with Independent Rate Allocation for Same Rate Streams

The results obtained from the simulations for sources with joint rate allocation and same rate streams are summarized in Table A.3.

Table A.3.: Overall throughput Θ for sources with independent rate allocation and same rate streams

C_{S_1}	C_{S_2}	Θ	X	Y
(kbps)	(kbps)	(kbps)	(kbps)	(kbps)
200	600	200	100	100
300	600	300	150	150
400	600	400	200	200
500	600	500	250	250
600	600	575	287.5	287.5
700	600	600	300	300
800	600	600	300	300
900	600	600	300	300

A.1.1.4. Sources with Independent Rate Allocation for Different Rate Streams

The results obtained from the simulations for sources with joint rate allocation and same rate streams are summarized in Table A.4.

C_{S_1}	C_{S_2}	Θ	X	Y
(kbps)	(kbps)	(kbps)	(kbps)	(kbps)
200	600	337.5	100	237.5
300	600	400	150	250
400	600	462.5	200	262.5
500	600	525	250	275
600	600	575	287.5	287.5
700	600	600	300	300
800	600	612.5	312.5	300
900	600	625	325	300

Table A.4.: Overall throughput Θ for sources with independent rate allocation and different rate streams

A.1.2. Results for PSNR-based Simulation

Scenarios for simulation based on video quality (PSNR):

- 1. Sources with joint rate allocation for different video quality streams,
- 2. Sources with joint rate allocation for same video quality streams,
- 3. Sources with independent rate allocation for same rate streams,
- 4. Sources with independent rate allocation for different video quality streams.

A.1.2.1. Sources with Joint Rate Allocation for Different Video Quality Streams

The resulting *PSNR* from the simulations for sources with joint rate allocation and different rate streams are summarized in Table A.5.

C_{S_1}	C_{S_2}	PSNR	$PSNR_1$	$PSNR_2$	X	Y	Θ
(kbps)	(kbps)	(dB)	(dB)	(dB)	(kbps)	(kbps)	(kbps)
200	600	79.05	36.49	42.56	200	275	475
300	600	79.73	37.84	41.89	265.5	234.5	500
400	600	80.17	38.10	42.07	280.2	244.8	525
500	600	80.59	38.35	42.24	295	255	550
600	600	80.99	38.58	42.41	309.7	265.3	575
700	600	81.38	38.81	42.57	324.6	275.4	600
800	600	81.75	39.03	42.72	339.6	285.4	625
900	600	82.10	39.25	42.85	354.7	295.3	650

Table A.5.: PSNR for sources with joint rate allocation and different video quality streams

Although the best overall throughput can be achieved and the upload capacity of both sources and requesting peers is fully exhausted, the sequences *X* and *Y* have different rates.

A.1.2.2. Sources with Joint Rate Allocation for Same Video Quality Streams

The resulting PSNR from the simulations for sources with joint rate allocation and same rate streams are summarized in Table A.6. The resulting PSNRs for video sequences X and Y are very different.

C_{S_1}	C_{S_2}	PSNR	$PSNR_1$	$PSNR_2$	X	Y	Θ
(kbps)	(kbps)	(dB)	(dB)	(dB)	(kbps)	(kbps)	(kbps)
200	600	77.70	36.49	41.21	200	200	400
300	600	79.71	37.55	42.16	250	250	500
400	600	80.15	37.78	42.36	262.5	262.5	525
500	600	80.56	38.00	42.56	275	275	550
600	600	80.97	38.22	42.75	287.5	287.5	575
700	600	81.35	38.43	42.92	300	300	600
800	600	81.72	38.63	43.09	312.5	312.5	625
900	600	82.07	38.82	43.25	325	325	650

Table A.6.: PSNR for sources with joint rate allocation and same rate streams

On the other hand, Table A.7 shows the resulting $PSNR_e$ after the PSNR enforcement have been made. Videos sequences with a similar PSNR are obtained, although their individual rates are different. The results show that when the source S_1 is 200 kbps, the PSNR enforcement is not possible, because the maximum rate reached by sequence X is identical to the upload capacity of S_1 .

C_{S_1}	C_{S_2}	$PSNR_e$	$PSNR_{e1}$	$PSNR_{e2}$	X	Y	Θ
(kbps)	(kbps)	(dB)	(dB)	(dB)	(kbps)	(kbps)	(kbps)
200	600	79.05	36.49	42.56	200	275	475
300	600	79.66	38.44	41.21	300	200	500
400	600	79.46	39.54	39.91	375.33	149.66	525
500	600	79.80	39.86	39.94	399.5	150.5	550
600	600	80.27	40.05	40.22	415	160	575
700	600	80.77	40.18	40.59	426.33	173.67	600
800	600	81.22	40.34	40.88	439.33	185.67	625
900	600	81.53	40.60	40.93	462.33	187.66	650

Table A.7.: PSNR for sources with joint rate allocation after the PSNR enforcement

A.1.2.3. Sources with Independent Rate Allocation for Same Rate Streams

The resulting *PSNR* from the simulations for sources with independent rate allocation and same rate streams are summarized in Table A.8.
C_{S_1}	C_{S_2}	PSNR	$PSNR_1$	$PSNR_2$	X	Y	Θ
(kbps)	(kbps)	(dB)	(dB)	(dB)	(kbps)	(kbps)	(kbps)
200	600	71.08	33.13	37.95	100	100	200
300	600	75.06	35.12	39.93	150	150	300
400	600	77.72	36.49	41.22	200	200	400
500	600	79.73	37.55	42.16	250	250	500
600	600	80.98	38.24	42.74	287.5	287.5	575
700	600	81.36	38.43	42.93	300	300	600
800	600	81.36	38.43	42.92	300	300	600
900	600	81.36	38.43	42.92	300	300	600

Table A.8.: PSNR for independent sources with same rate streams

A.1.2.4. Sources with Independent Rate Allocation for Different Video Quality Streams

The resulting *PSNR* from the simulations for sources with independent rate allocation and different video quality streams are summarized in Table A.9.

C_{S_1}	C_{S_2}	PSNR	$PSNR_1$	$PSNR_2$	X	Y	Θ
(kbps)	(kbps)	dB	dB	(dB)	(kbps)	(kbps)	(kbps)
200	600	75.11	33.16	41.95	100	237.5	237.5
300	600	77.30	35.14	42.16	150	200	350
400	600	78.87	36.51	42.36	200	262.5	462.5
500	600	80.13	37.57	42.56	250	275	525
600	600	80.98	38.24	42.74	287.5	287.5	575
700	600	81.36	38.44	42.92	300	300	600
800	600	81.56	38.64	42.92	312.5	300	612.5
900	600	81.75	38.83	42.92	325	300	625

Table A.9.: PSNR for independent sources with different video quality streams

A.2. Experiments based on Scalable Video Coding

This section presents the results from enforcing the same video quality for two different video sequences introduced in Chapter 5. In this case, scalable video coding is used in the adaptive multi-source scheme to achieve an identical video quality for both sequences. The Foreman and Daughter and Mother sequences are used as test sequences, and both are encoded using the JSVM software version 9.1 [182].

Mother and Daughter sequence (Figure A.1) is captured with a fixed camera and shows two persons in the foreground. A person is talking, while the other persona is fixed. The encoding uses these parameters:



Figure A.1.: Example picture of the Mother and Daughter video sequence.

- Spatial resolution: CIF (352 x 288)
- Temporal resolution: 30 frames per second
- Number of frames: 3000 frames

Table A.10 shows the resulting PSNR and rate for this sequence when one base layer (BL) and one enhancement layer (EL) are encoded.

 			8	(
Layer	Rate $(kbps)$	Y-PSNR (dB)	U-PSNR (dB)	V-PSNR (dB)
0	98.57	39.8212	44.5891	45.4647
1	285.35	42.0363	45.5846	46.6068

Table A.10.: PSNR and rate for Mother & Daughter sequence (1 BL and 1 EL)

Foreman sequences (Figure A.2) is captured by a mobile device, and shows a man talking in a construction site. The encoding uses these parameters:

- Spatial resolution: CIF (352 x 288)
- Temporal resolution: 30 frames per second
- Number of frames: 3000 frames

First, the Foreman sequence is encoded using one base layer and one enhancement layer only. The resulting PSNR and rate are shown in Table A.11.

Layer	Rate (kbps)	Y-PSNR(dB)	U-PSNR (dB)	V-PSNR (dB)
0	273.9408	36.3277	41.3289	43.0568
1	627.4880	39.0984	43.6449	46.2311

Table A.11.: PSNR and rate for Foreman sequence (1 BL and 1 EL)



Figure A.2.: Example picture of the Foreman video sequence.

The resulting PSNR is smaller than the resulting PSNR from the Mother & Daughter sequence. Then a second enhancement layer must be encoded. Table A.12 shows the resulting PSNR and rate for the Foreman video sequence when one base layers and two enhancement layer are encoded.

Layer	Rate (kbps)	Y-PSNR (dB)	U-PSNR (dB)	V-PSNR (dB)
0	273.9408	36.3277	41.3289	43.0568
1	627.4880	39.0984	43.6449	46.2311
2	1501.8048	42.0139	45.6061	47.8623

 Table A.12.: PSNR and rate for Foreman sequence (1 BL and 2 EL)

Now, both video sequences have a very similar PSNR.

A. Appendix

Bibliography

- [1] F. Fluckiger, "Understanding Networked Multimedia, applications and technology," Prentice Hall 1995.
- [2] Napster Inc. Napster homepage. http://www.napster.com/, seen on 02.2008.
- [3] Gnutella 0.4. http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf, seen on 02.2008.
- [4] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnana, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications," in the Proc. of the ACM SIGCOMM 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, pp. 149-160, San Diego, CA, USA, August 2001.
- [5] KaZaA file sharing network. KaZaA homepage. http://www.kazaa.com/, seen on 02.2008.
- [6] Global IP Traffic Forecast and Methodology, 2006-2011, Cisco White paper, updated January 14, 2008.
- [7] D. S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, B. Richard, S. Rolling, and Z. Xu, "Peer-to-Peer Computing," Technical Report HPL-2002-57R1, HP Laboratories, 2002.
- [8] B. Cohen, "Incentives Build Robustness in BitTorrent," in the Proc. of the 1st Workshop on Economics of Peer-to-Peer Systems, Berkeley, CA, USA, June 2003.
- [9] S. Androutsellis-Theotokis and D. Spinellis, "A Survey of Peer-to-Peer Content Distribution Technologies," in ACM Computing Surveys, Vol. 36, Num. 4, pp. 335-371, December 2004.
- [10] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. W. O'Toole Jr., "Overcast: Reliable Multicasting with an Overlay Network," in Proc. of the 4th Symposium on Operating System Design and Implementation (OSDI'00), pp. 197-212, San Diego, CA, USA, October 2000.
- [11] M. Ripeanu, I. Foster, A. Iamnitchi, and A. Rogers, "In Search for Simplicity: A Self-Organizing Multi-Source Multicast Overlay," in Proc. of the 1st IEEE International Conference (SASO'07), pp. 371-374, Boston, MA, USA, July 2007.
- [12] A. Rowstron and P. Druschel, "Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems," in Proc. of the IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), pp. 329-350, Heidelberg, Germany, November, 2001.
- [13] B. Y. Zhao, J. Kubiatowicz, and A. Joseph, "Tapestry: An Infrastructure for Fault-Tolerant Wide-Area Location and Routing," Technical Report UCB/CSD-01-1141, University of California at Berkeley, Computer Science Department, 2001.
- [14] S. Ratnassamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A Scalable Content-Addressable Network," in the Proc. of the ACM SIGCOMM 2001 Conference on Ap-

plications, Technologies, Architectures, and Protocols for Computer Communication, pp. 161-172, San Diego, CA, USA, August 2001.

- [15] J. Li and S. Vuong, "An Efficient Clustered Architecture for P2P Networks," in Proc. of the 18th International Conference on Advanced Information Networking and Applications (AINA'04), pp. 278-283, Vol.1, Fukuoka, Japan, March 2004.
- [16] M. Castro, M. Costa, and A. Rowstron, "Peer-to-Peer Overlays: structured, unstructured, or both?," Technical Report MSR-TR-2004-73, Microsoft Research, 2004.
- [17] Schollmeier Rüdiger, "Signaling and Networking in Unstructured Peer-to-Peer Networks," PhD Thesis, Technische Universität München (TUM), Munich, Germany, April 2005.
- [18] FastTrack Peer-to-Peer technology company. FastTrack homepage. http://www.fasttrack.nu/, seen on 02.2008.
- [19] The Free Network Project. http://freenetproject.org/, seen on 02.2008.
- [20] Skype homepage. http://www.skype.com/, seen on 02.2008.
- [21] JXTA homepage. http://www.jxta.org, seen on 02.2008.
- [22] Emule homepage. http://www.emule-project.net/home, seen on 02.2008.
- [23] Gnutella 0.6, "RFC-Gnutella 0.6". http://rfc-gnutella.sourceforge.net/src/rfc-0 _6draft.html, June 2002, seen on 02.2008.
- [24] R. Schollmeier, "A Definition of Peer-to-Peer Networking for the Classification of Peerto-Peer Architectures and Applications," in Proc. of the First International Conference on Peer-to-Peer Computing, pp. 101-102, Linköping, Sweden, August 2001.
- [25] J. De Boever, "Peer-to-Peer Networks as a Distribution and Publishing Model," in Proc. of the 11th International Conference on Electronic Publishing (ELPUB2007), pp. 175-188, Vienna, Austria, June 2007.
- [26] H. Shen, C. Z. Xu and G. Chen, "Cycloid: A Constant-Degree and Lookup-Efficient P2P Overlay Network," in Proc. of the 18th International Parallel and Distributed Processing Symposium (IPDPS'04), pp. 195-216, Santa Fe, NM, USA, April 2004.
- [27] W. Zheng, X. Liu, S. Shi, J. Hu, and H. Dong, "Peer-to-Peer: A Technique Perspective," in Handbook of Theoretical and Algorithmic Aspects of Ad Hoc, Sensor, and Peer-to-Peer Networks, Auerbach Publications, pp. 587-588, 2006.
- [28] D. A. Tran, K. A. Hua, and T. T. Do, "A Peer-to-Peer Architecture for Media Streaming," in IEEE Journal on Selected Areas in Communication, Special Issue on Advances in Overlay Networks, Vol.22 Num.1, pp. 121-133, January 2004.
- [29] S. Khan, R. Schollmeier, and E. Steinbach, "A Performance Comparison of Multiple Description Video Streaming in Peer-to-Peer and Content Delivery Networks," in Proc. of the IEEE International Conference on Multimedia and Expo (ICME'04), pp. 503-506, Taipei, Taiwan, June 2004.
- [30] K. Aberer, "P-grid: A self-organizing access structure for P2P information systems," in Proc. of the International Conference on Cooperative Information Systems, Vol. 2172, pp. 179-194, Trento, Italy, September 2001.
- [31] G. Moro, A. M. Ouksel, and C. Sartori, "Agents and Peer-to-Peer Computing: A Promising Combination of Paradigms," in Proc. of the Agents and Peer-to-Peer Computing, Vol. 2530/2008, pp. 15-28, Bologna, Italy, July 2002.

- [32] L. G. A. Sung, N. Ahmed, R. Blanco, H. Li, M. A. Soliman, and D. Hadaller, "A Survey of Data Management in Peer-to-Peer Systems," Technical Report CS-2006-18, University of Waterloo, June 2006.
- [33] D. Barkai, "Technologies for Sharing and Collaborating on the Net," in Proc. of the First International Conference on Peer-to-Peer Computing, pp. 13-28, Linköping, Sweden, August 2001.
- [34] I. Foster, "The Grid: A New Infrastructure for 21st Century Science," Physics Today 55, 42-6, 2002.
- [35] Genome@home homepage. http://genomeathome.stanford.edu/, seen on 02.2008.
- [36] S. M. Larson, C. D. Snow, M. Shirts, and V. S. Pande, "Folding@Home and Genome@Home: Using distributed computing to tackle previously intractable problems in computational biology," in Computational Genomics, Richard Grant, editor, Horizon Press 2003.
- [37] BOINC homepage. http://boinc.berkeley.edu/, seen 02.2008.
- [38] SETI@home homepage. http://setiathome.ssl.berkeley.edu/, seen on 02.2008.
- [39] D. P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer, "SETI@home: an experiment in public-resource computing," in Communications of the ACM, Vol. 45, Issue 11, pp. 56-61, 2002.
- [40] W. T. Sullivan, III, D. Werthimer, S. Bowyer, J. Cobb, D. Gedye, and D. Anderson, "A new major SETI project based on project Serendip data and 100,000 personal computers," in Proc. of the 5th International Conference on Bioastronomy IAU Colloquium No. 161. Capri, July 1996.
- [41] D. P. Anderson, "BOINC: A System for Public-Resource Computing and Storage," in Proc. of the Fifth IEEE/ACM International Workshop on Grid Computing (GRID'04), pp. 4-10, Pittsburgh, PA, USA, November 2004.
- [42] V. Berstis, "Fundamentals of Grid Computing," IBM, 2003.
- [43] I. Foster and A. Iamnitchi, "On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing," in Proc. of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03), pp. 118-128, Berkeley, CA, USA, February 2003.
- [44] Microsoft homepage. http://www.microsoft.com/technet/technetmag/issues/2006/ 10/IntoTheGroove/
- [45] BSCW homepage. http://public.bscw.de/bscw_help-4.3/english/
- [46] Oculus Technology homepage. http://www.oculustech.com/co/co-techoverview.pdf
- [47] Oculus Technology, "CO Technical Overview," White paper, 2001.
- [48] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai, "Distributing Streaming Media Content Using Cooperative Networking," in Proc. of the 12th ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'02), pp. 177-186, Miami, Florida, USA, May 2002.
- [49] J. Li, P. A. Chou, and C. Zhang, "Mutualcast: An Efficient Mechanism for One-To-Many Content Distribution," in Proc. of the ACM SIGCOMM ASIA Workshop, Beijing, China, April 2005.

- [50] F. A. López-Fuentes and E. Steinbach, "Collaborative Hierarchical Multicast," in Proc. of the 15th ACM Multimedia Conference (ACM MM'07), pp. 763-766, Ausburg, Germany, September 2007.
- [51] X. Zhang, J. Liu, B. Li, and T.-S. P. Yum, "CoolStreaming/DONet: A Data-driven Overlay Network for Efficient Live Media Streaming," in Proc of the IEEE 24th INFOCOM, Vol. 3, pp. 2102- 2111, Miami, FL, USA, March 2005.
- [52] Sopcast homepage. http://www.sopcast.com, seen on 02.2008
- [53] PPLive homepage. http://www.pplive.com/en/index.html, seen on 02.2008
- [54] PPstream homepage. http://www.ppstream.cn/, seen on 02.2008
- [55] W. T. Ooi, "Dagster: Contributor Aware End-Host Multicast for Media Streaming in Heterogeneous Environment," in Proc. of the ACM/SPIE Multimedia Computing and Networking Conference, Vol. 5680, pp. 77-90, San Jose CA, January 2005.
- [56] Y. Guo, K. Suh, J. Kurose, and D. Towsley, "P2Cast: Peer-to-peer Patching Scheme for VoD Service," in Proc. of the 12th international conference on World Wide Web, pp. 301-309, Budapest, Hungary, May 2003.
- [57] X. Jiang, Y. Dong, D. Xu, and B. Bhargava, "GnuStream: A P2P Media Streaming System Prototype," in Proc. of the International Conference on Multimedia and Expo (ICME'03), pp. 325-328, Baltimore, MD, USA, July 2003.
- [58] W. Gao and L. Huo, "Challenges on Peer-to-Peer Live Media Streaming," in Proc. of the International Workshop on Multimedia Content Analysis and Mining (MCAM'07), Vol. 4577/2007, pp. 37-41, Weihai, China, June 2007.
- [59] A. B. Bondi, "Characteristics of Scalability and Their Impact on Performance," in Proc. of the Workshop on Software and Performance, pp. 195-203, Ottawa, Ontario, Canada, September 2000.
- [60] A. Pfitzmann and M. Waidner, "Networks Without User Observability Design Options," in Proc. of the EUROCRYPT 1985, pp. 245-253, Linz, Austria, April 1985.
- [61] C. Gershenson, "Design and Control of Self-Organizing Systems," PhD thesis, Vrije Universiteit Brussel, Brussels, Belgium, May 2007.
- [62] C. Prehofer and C. Bettstetter, "Self-Organization in Communication Networks: Principles and Design Paradigms," in IEEE Communications Magazine, Feature Topic on Advances in Self-Organizing Networks, Vol. 43, Num. 7, pp. 78-85, July 2005.
- [63] J. Ledlie, J. M. Taylor, L. Serban, and M. Seltzer, "Self-Organization in Peer-to-Peer Systems," in Proc. of the 10th workshop on ACM SIGOPS European workshop, pp. 125-132, Saint-Emilion, France, July 2000.
- [64] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, R. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao, "OceanStore: An Architecture for Global-Scale Persistent Storage," in Proc. of the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems (ASP-LOS'00), pp. 190-201, Cambridge, MA, USA, November 2000.
- [65] E. Cohen and S. Shenker, "Replication Strategies in Unstructured Peer-to-Peer Networks," in Proc. of the ACM SIGCOMM'02, pp. 177-190, Pittsburgh, PA, USA, October 2002.
- [66] B. Bhattacharjee, S. Chawathe, V. Gopalakrishnan, P. Keleher, and B. Silaghi, "Efficient

Peer-To-Peer Searches Using Result-Caching," in Proc. of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03), Vol. 2735/2003, pp. 225-236, Berkeley, CA, USA, February 2003.

- [67] B. Yang and H. Garcia-Molina, "Comparing Hybrid Peer-to-Peer Systems," The VLDB Journal, pp. 561-570, September 2001.
- [68] M. K. Ramanathan, V. Kalogeraki, and J. Pruyne, "Finding Good Peers in the Peerto-Peer Networks," in Proc. of the International Parallel and Distributed Computing Symposium (IPDPS'02), pp. 24-31, Ft. Lauderdale, FL, USA, April 2002.
- [69] M. Li, W.-C. Lee, and A. Sivasubramaniam, "Semantic Small World: An Overlay Network for Peer-to-Peer Search," in Proc. of the 12th IEEE International Conference on Network Protocols (ICNP'04), pp. 228-238, Berlin, Germany, October 2004.
- [70] S. Jagannathan and G. Pandurangan, "Stochastic Analysis of a Fault-Tolerant and Bandwidth-Efficient P2P Network," Technical Report CSD TR-03-029, Purdue University, 2003.
- [71] A. Grimshaw and W. A. Wulf, "The Legion Vision of a Worldwide Virtual Computer," in Communications of the ACM, Vol. 40, Num. 1, pp. 39-45, January 1997.
- [72] C. Huitema, "Routing in the Internet," 1st edition, Prentic-Hall, New Jersey, 1995, 319 pages.
- [73] J. G. Apostolopoulos, W. T. Tan, and S. J. Wee, "Video streaming: Concepts, Algorithms, and Systems," Technical report HPL-2002-260, HP Laboratories Palo Alto, 2002.
- [74] D. Wu, Y. T. Hou, W. Zhu, Y.-Q. Zhang, and J. M. Peha, "Streaming Video over the Internet: Approaches and Directions," in IEEE Transactions on Circuits and Systems for Video Technology, Vol. 11, Num. 3, pp. 282-300, March 2001.
- [75] Y. Wang, J. Ostermann, and Y.-Q. Zhang, "Video Processing and Communications," Prentice Hill, 2002.
- [76] F. Halsall, "Multimedia Communications, Applications, Networks, Protocols and Standards," Addison Wesley, 2001.
- [77] Y. Chu, S. G. Rao, S. Seshan, and H. Zhang, "Enabling Conferencing Applications on the Internet using an Overlay Multicast Architecture," in Proc. of the ACM SIG-COMM'01, pp. 55-67, San Diego, CA, USA, August 2001.
- [78] L. Kontothanassisy, R. Sitaramanz, J. Weinz, D. Hongz, R. Kleinberg, B. Mancusoz, D. Shawz, and D. Stodolsky, "A Transport Layer for Live Streaming in a Content Delivery Network," in Proc. of the IEEE, Vol. 92, Num. 9, pp. 1408-1419, September 2004.
- [79] J. A. T. Kangasharju, "Internet Content Distribution," PhD Thesis, Nice Sophia Antipolis University, Nice, France, April 2002.
- [80] E. Setton, "Congestion-aware Video Streaming over Peer-to-Peer Networks," PhD Thesis, Stanford University, Stanford, CA, USA, December 2006.
- [81] S. E. Deering, "Multicast Routing in Internetwork and Extended LANs," in Proc. of the ACM SIGCOMM'88, pp. 55-64, Stanford, CA, USA, August 1988.
- [82] E. Setton, P. Baccichet, and B. Girod, "Peer-to-Peer Live Multicast: A Video Perspective," in Proc. of the IEEE, Vol. 96, Num. 1, pp. 25-38, January 2008.
- [83] Cisco "Internetworking Technologies Handbook," Chapter 43, pp. 1-16.

- [84] B. Zhang, W. Wang, S. Jamin, D. Massey, and L. Zhang, "Universal IP Multicast Delivery," in Computer Networks, Elsevier, Vol. 50, pp. 781-806, 2006.
- [85] D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel, "ALMI: An Application Level Multicast Infrastructure," in Proc. of the Third USENIX Symposium on Internet Technologies and Systems (USITS'01), pp. 49-60, San Francisco, CA, USA, March 2001.
- [86] M. Brogle, D. Milic, and T. Braun, "Supporting IP Multicast Streaming Using Overlay Networks," in Proc. of the International Conference on Heterogeneous Network for Quality, Reliability, Security and Robustness, ACM Qshine'07, Vancouver, Canada, August 2007.
- [87] X. Jin, K.-L. Cheng, and S.-H. G. Chan, "Scalable Island Multicast for Peer-to-Peer Streaming," in Advances in Multimedia vol. 2007, Article ID 78913, pp. 1-9, Hindawi Publishing Corporation.
- [88] A. Fei, J. Cui, M. Gerla, and M. Faloutsos, "Aggregated Multicast: an Approach to Reduce Multicast State," in Proc. of the 6th Global Internet Symposium in Conjunction with GLOBECOM'01, Vol. 3, pp. 1595-1599, San Antonio, TX, USA, November 2001.
- [89] L. H. M. K. Costa, S. Fdida, and O.C.M.B. Duarte, "Hop by hop Multicast Routing Protocol," in Proc. of the ACM SIGCOMM'01, pp. 249-259, San Diego, CA, USA, August 2001.
- [90] I. Stoica, T. S. E. Ng, and H. Zhang, "REUNITE: A Recursive Unicast Approach to Multicast," in Proc. of the IEEE INFOCOM 2000, Vol. 3, pp. 1644-1653, Tel-Aviv, Israel, March 2000.
- [91] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable Application Layer Multicast," in Proc. of the ACM SIGCOMM'02, pp. 205-217, Pittsburgh, PA, USA, August 2002.
- [92] Y. Chu, S. G. Rao, S. Seshan, and H. Zhang, "A Case for End System Multicast," in IEEE Journal on Selected Areas in Communications, Vol. 20, Num. 8, pp. 1456-1471, October 2002.
- [93] F. A. López-Fuentes and E. Steinbach, "Multi-Source Video Multicast in Peer-to-Peer Networks," In Proc. of the Fifth International Workshop on Hot Topics in Peer-to-Peer Systems (HotP2P'07) held in conjunction with the 22nd IEEE International Parallel and Distributed Processing Symposium (IPDPS'08), pp. 1-8, Miami, FL, USA, April 2008.
- [94] T. Nguyen and A. Zakhor, "Multiple Sender Distributed Video Streaming," in IEEE Transactions on Multimedia, Vol. 6, Num. 2, pp. 315-326, April 2004.
- [95] N. Magharei and R. Rejaie, "Understanding Mesh based Peer to Peer Streaming," in Proc. of the 16th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'06), Newport, RI, USA, May 2006.
- [96] Z. Liu, H. Yu, D. Kundur, and M. Merabti, "On Peer-to-Peer Multimedia Content Access and Distribution," in Proc. of the IEEE International Conference on Multimedia and Expo (ICME'06), pp. 557-560, Toronto, Ontario, CA, July 2006.
- [97] N. Magharei, R. Rejaie, and Y. Guo, "Mesh or Multiple-Tree: A Comparative Study of Live P2P Streaming Approaches," in Proc. of the IEEE INFOCOM 2007, pp. 1424-1432, Anchorage, Alaska, USA, May 2007.
- [98] Y. Zhu and B. Li, "Correlation-aware Multimedia Content Distribution in Overlay Net-

works," in Proc. of the 30th Annual SPIE / ACM Conference on Multimedia Computing and Networking, pp. 127-138, San Jose, CA, USA, January 2006.

- [99] J. Li, Y. Cui, and B. Chang, "PeerStreaming: design and implementation of an ondemand distributed streaming system with digital rights management capabilities," in Multimedia Systems, Springer, Vol. 13, Num. 3, pp. 173-190, September 2007.
- [100] G. Wang, S. Futemma, and E. Itakura, "Multiple Description Coding for Overlay Network Streaming," in IEEE MultiMedia Volume 14, Issue 1, pp. 74-82, January 2007.
- [101] S. Veńot and L. Yang, "Peer-to-Peer Media Streaming Application Survey," in Proc. of the International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM 2007), pp. 139-148, Papeete, French Polynesia, November 2007.
- [102] E. Setton, J. Noh, and B. Girod, "Rate-Distortion Optimized Video Peer-to-Peer Multicast Streaming," in Proc. of the Workshop on Advances in Peer-to-Peer Multimedia Streaming at ACM Multimedia'05, pp. 39-48, Singapore, Singapore, November 2005.
- [103] M. Hefeeda, A. Habib, D. Xu, B. Bhagava, and B. Botev, "Collectcast: A Peer-to-Peer Service for Media Streaming," in ACM/Springer Multimedia System Journal, Vol. 11, Num. 5, pp. 68-81, November 2005.
- [104] L. Choi, M. Ivrlac, E. Steinbach, and J. Nossek, "Sequence-level Models for Distortion-Rate Behaviour," in Proc. of the IEEE ICIP 2005, Vol. 2, pp. 486-489, Genova, Italy, September 2005.
- [105] H.264/MPEG-4 AVC Reference Software http://iphone.hhi.de/suering/tml/
- [106] M. Burger, T. Kielmann, and H. E. Bal, "Balanced Multicasting: High-throughput Communication for Grid Applications," in Proc. of the ACM/IEEE Supercomputing, Seattle, WA, USA, November 2005.
- [107] S. G. Nash and A. Sofer, "Linear and Nonlinear Programming," McGraw-Hill International Editions 1996.
- [108] N. Magharei and R. Rejaie, "Adaptive Receiver-Driven Streaming from Multiple Senders," in ACM/Springer Multimedia Systems Journal 11, Num. 6, pp. 1-18, 2006.
- [109] F. Wang, Y. Xiong, and J. Liu, "mTreebone: A Hybrid Tree/Mesh Overlay for Application-Layer Live Video Multicast," in Proc. of the 27th International Conference on Distributed Computing Systems (ICDCS'07), pp. 49-56, Toronto, Ontario, Canada, June 2007.
- [110] M. Castro, P. Druschel, A-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "Split-Stream: High-Bandwidth Content Distribution in Cooperative Environments," in Proc. of the 19th ACM Symposium on Operating System Principles, pp. 298-313, Bolton Landing, NY, USA, October 2003.
- [111] D. Carra, R. Lo Cigno, and E. W. Biersack, "Graph Based Analysis of Mesh Overlay Streaming Systems," in IEEE Journal on Selected Areas in Communications, Vol. 25, Num. 9, pp. 1667-1677, December 2007.
- [112] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat, "Bullet: High Bandwidth Data Dissemination Using and Overlay Mesh," in Proc. of the 19th ACM Symposium on Operating Systems Principles, pp. 282-297, Bolton Landing, NY, USA, October 2003.
- [113] S. Zhuang, B. Zhao, A. Joseph, R. Katz, and J. Kubiatowicz, "Bayeux: An Architecture

for Scalable and Fault-Tolerant Wide-Area Data Dissemination," in Proc. of the 11th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'01), pp. 11-20, Port Jefferson, NY, USA, June 2001.

- [114] F. Pianese, D. Perino, J. Keller, and E. W. Biersack, "PULSE: an Adaptive, Incentivebased, Unstructured P2P Live Streaming System," in IEEE Transactions on Multimedia, Special Issue on Content Storage and Delivery in Peer-to-Peer Networks, Vol. 9, Num. 8, pp. 1645-1660, December 2007.
- [115] M. Castro, P. Druschel, A. -M. Kermarrec, and A. Rowstron, "SCRIBE: A large-scale and decentralized application-level multicast infrastructure," in IEEE Journal on Selected Areas in Communications, Vol. 20, Num. 8, pp. 1489- 1499, October 2002.
- [116] X. Hei, Y. Liu, and K. W. Ross, "Inferring Network-Wide Quality in P2P Live Streaming Systems," in IEEE Journal on Selected Areas in Communications, Vol. 25, Num. 9, pp. 1640-1654, December 2007.
- [117] W. -P. K. Yiu, X. Jin, and S.-H. G. Chan, "VMesh: Distributed Segment Storage for Peer-to-Peer Interactive Video Streaming," in IEEE Journal on Selected Areas in Communications, Vol. 25, Num. 9, pp. 1717-1731, December 2007.
- [118] G. Wu and T. Chiueh, "How efficient is BitTorrent?," in Proc. of the 30th ACM/SPIE Multimedia Computing Network (MMCN'06), San Jose, CA, USA, January 2006.
- [119] X. Su and T. Wang, "Sequence of Linear Programming for Transmission of Fine-Scalable Coded Content in Bandwidth-Limited Environments," in Multimedia Systems, Vol. 11, Num. 5, pp. 455-466, Springer Verlag, 2006.
- [120] C. Hsu and M. Hefeeda, "Optimal Bit Allocation for Fine-grained Scalable Video Sequences in Distributed Streaming Environments," in Proc. of the 40th ACM/SPIE Multimedia Computing and Networking (MMCN'07), San Jose, CA, USA, 2007.
- [121] S. Pemmaraju and S. Skiena, "Computational Discrete Mathematics: Combinatorics and Graph Theory with Mathematica," Cambridge University Press, 2003.
- [122] S. -W. Tan, G. Waters, and J. Crawford, "MeshTree: Reliable Low Delay Degreebounded Multicast Overlays," in Proc. of the 11th International Conference on Parallel and Distributed Systems (ICPADS'05), Vol. 2, pp. 565- 569, Fukuoka, Japan, July, 2005.
- [123] F. A. López-Fuentes and E. Steinbach, "Proximity-Aware Collaborative Multicast for Small P2P Communities," In Proc. of the Fourth International Workshop on Hot Topics in Peer-to-Peer Systems (HotP2P'07) held in conjunction with the 21th IEEE International Parallel and Distributed Processing Symposium (IPDPS'07), pp. 1-8, Long Beach, CA, USA, March 2007.
- [124] F. A. López-Fuentes and E. Steinbach, "Architecture for Media Streaming Delivery over P2P Networks," in Proc. of the IEEE Fifth International Symposium and School on Advanced Distributed Systems (ISSADS'05), Vol. 3563/2005, pp. 72-82, Guadalajara, Jalisco, Mexico, January 2005.
- [125] D. Jurca, J. Chakareski, J. -P. Wagner, and P. Frossard, "Enabling Adaptive Video Streaming in P2P Systems," in IEEE Communications Magazine, Vol. 45, Num. 6, pp. 108-114, June 2007.
- [126] Y. Cui, B. Li, and K. Nahrstedt, "On Achieving Optimized Capacity Utilization in Application Overlay Networks with Multiple Competing Sessions," in Proc. of the ACM

16th Symposium on Parallelism in Algorithms and Architectures (SPAA'04), pp. 160-169, Barcelona, Spain, June 2004.

- [127] H. Prüfer, "Neu Beweis eines Satzes über Permutationen," in Archiv für Mathematik und Physik, Vol. 27, pp. 142-144, 1918.
- [128] F. A. López-Fuentes and E. Steinbach, "Collaborative Content Distribution with Multipple Coexisting Sessions," Technical Report LKN-MTG-TR-04, München Technical Universität, March, 2006.
- [129] W. K. Grassmann and J. P.Tremblay, "Logical and Discrete Mathematics: A Computer Science Perspective," Prentice Hall, 1996.
- [130] B. R. Preiss, "Data Structures and Algorithms with Object-Oriented Design Patterns in Java," John Wiley & Sons, 1999.
- [131] J. A. Fill and R. P. Dobrow, "The Number of m-ary Search Trees on *n* Keys," in Combinatorics, Probability and Computing, Vol. 6, pp. 435-453, 1997.
- [132] A. Cayley, "A Theorem on Trees," Quart. J. Math. 23, pp. 376-378, 1889.
- [133] J. L. Gross and J. Yellen, "Graph Theory and its applications," Second Edition, Chapman & Hall/CRC.
- [134] I. M. Gessel and S. Seo, "A Refinement of Cayley's Formula for Trees," in the Electronic Journal of Combinatorics, Vol. 11, Num. 2, pp. 1-23, February 2006.
- [135] T. Koshy, "Discrete Mathematics with Applications," Second Edition, Elsevier Academic Press, 2007.
- [136] F. L. Heller and A. L. Tharp, "The *M-ary Tree and *Ternary Hillsort," in Proc. of the ACM 20th Annual Conference on Computer Science, pp. 41-48, Kansas City, MO, USA, March 1992.
- [137] L. Devroye, "On the height of random m-ary search trees," in Random Structures and Algorithms, Vol. 1, pp. 191-203, 1990.
- [138] B. Chauvin and N. Pouyanne, "m-ary Search Trees when m>26: A Strong Asymptotics for the Space Requirements" in Random Structures and Algorithms, Vol 24, Num. 2, pp. 133-154, 2004.
- [139] J. Li, "MutualCast: A Serverless Peer-to-Peer Multiparty Real-Time Audio Conferencing System," in Proc. of the IEEE International Conference on Multimedia & Expo (ICME'05), pp. 602-605, Amsterdam, The Netherlands, July 2005.
- [140] C. Huang, P. A. Chou, J. Li, and C. Zhang, "Adaptive Peer-to-Peer Streaming with MutualCast," in Journal of Zhejiang University SCIENCE A, Vol. 7, Num. 5, pp. 737-748, April 2006.
- [141] Y. Chawathe, "Scattercast: An Adaptable Broadcast Distribution Framework," in ACM Multimedia Systems Journal Special Issues on Multimedia Distribution, Vol. 9, Num. 1, pp. 104-118, July 2003.
- [142] A. R. Bharambe, S. G. Rao, V. N. Padmanabhan, S. Seshan, and H. Zhang, "The Impact of Heterogeneous Bandwidth Constraints on DHT Based Multicast Protocols," in Proc. of the 4th International Workshop on Peer-to-Peer Systems (IPTPS'05), Vol. 3640/2005, pp. 115-126, Ithaca, NY, USA, February 2005.
- [143] M. Ripeanu, I. Foster, and A. Iamnitchi, "Mapping the Gnutella Network: Properties

of Large-Scale Peer-to-Peer Systems and Implications for System Design," in Internet Computing Journal, Vol. 6, pp. 50-57, September 2002.

- [144] M. Hefeeda, A. Habib, B. Botev, D. Xu, and B. Bhargava, "PROMISE: Peer-to-Peer Media Streaming Using CollectCast," in Proc. of the ACM Multimedia'03, pp. 45-54, Berkeley, CA, USA, November 2003.
- [145] H. Pucha, D.G. Andersen, and M. Kaminsky, "Exploiting Similarity for Multi-Source Downloads Using File Handprints," in Proc. of the 4th USENIX NSDI'07, pp. 15-28, Cambridge, MA, April 2007.
- [146] N. Magharei and R. Rejaie, "Adaptive Receiver-driven Streaming from Multiple Senders," in ACM/Springer Multimedia Systems Journal 11, Num. 6, pp. 1-18, Juni 2006.
- [147] Y. Cui and K. Klara Nahrstedt, "Layered Peer-to-Peer Streaming," in Proc. of the 13th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'03), pp. 162-171, Monterey, California, June 2003.
- [148] R. Cohen and G. Kempfer, "A Unicast-based Approach for Streaming Multicast" in Proc. of the IEEE INFOCOM 2001, Vol 1, pp. 440-448, Anchorage, Alaska, USA, April 2001.
- [149] M. S. Kim, S.S. Lam, and D.Y. Lee, "Optimal Distribution Tree for Internet Streaming Media," in Proc. of the 23rd International Conference on Distributed Computing Systems (ICDCS'03), Providence, RI, USA, May 2003.
- [150] S. W. Tan, G. Waters, and J. Crawford, "A Survey and Performance Evaluation of Scalable Tree-based Application Layer Multicast Protocols," Technical Report No. 9-03, University of Kent, 2003.
- [151] A. El-Sayed, V. Roca, and L. Marthy, "A Survey of Proposals for an Alternative Group Communication Service," in IEEE Network, Vol. 17, Issue 1, pp. 46-51, January/February 2003.
- [152] J. Liu, B. Li, and Y-Q. Zhang, "Adaptive Video Multicast over the Internet," in IEEE Multimedia, Vol. 10, Issue 1, pp. 22-33, January/March 2003.
- [153] PlanetLab homepage. http://planet-lab.org, seen on 03.2008
- [154] L. Peterson, T. Anderson, D. Culler, and T. Roscoe, "A Blueprint for Introducing Disruptive Technology into the Internet," in ACM SIGCOMM Computer Communication Review, Vol. 33, Issue 1, pp. 59-64, January 2003.
- [155] A. Bavier, M. Bowman, B. Chun, D. Culler, S. Karlin, S. Muir, L. Peterson, T. Roscoe, T. Spalink, and M. Wawrzoniak, "Operating System Support for Planetary-Scale Network Services," in Proc. of the First Symposium on Networked Systems Design and Implementation (NSDI'04), San Francisco, California, March 2004.
- [156] L. Peterson, A. Bavier, M. Fiuczynski, and S. Muir, "Experiences Implementing PlanetLab," in Proc. of the 7th Symposium on Operating Systems Design and Implementation (OSDI'06), Seattle, WA, USA, November 2006.
- [157] F. A. López-Fuentes and E. Steinbach, "Adaptive Multi-source Video Multicast," in Proc. of the IEEE International Conference on Multimedia & Expo (ICME'08), pp. 457-460, Hannover, Germany, June 2008
- [158] L. Wang, K. S. Park, R. Pang, V. S. Pai, and L. Peterson, "Reliability and Security in

the CoDeeN Content Distribution Network," in Proc. of the USENIX'04, Boston, MA, USA, June-July 2004.

- [159] S. Ganguly, A. Saxena, S. Bhatnagar, and S. Banerjee, "Fast Replication in Content Distribution Overlays," in Proc. of the IEEE INFOCOM'05, Vol. 4, pp. 2246-2256, Miami, FL, USA, March, 2005.
- [160] One-Lab homepage. http://www.one-lab-2.org/ seen on 04.2008.
- [161] S. Y. Lee, J. G. Lee, and C. Y. Choi, "Multi-Source Media Streaming for the Contents Distribution in a P2P Network," in Proc. of the 5th Pacific Rim Conference on Multimedia (PCM'04), Vol. 3333/2005, pp. 290-297, Tokyo, Japan, November-December 2004.
- [162] S. Birrer, D. Lu, F. Bustamante, Y. Qiao, and P. Dinda, "FatNemo: Building a Resilient Multi-source Multicast Fat-Tree," in Proc. of the Workshop on Web Content Caching and Distribution, Vol. 3293/2004, pp. 182-196, Beijing, China, October 2004.
- [163] J. Jia and Q. Zhang, "Shared Tree for Application-layer Multi-Source Multicast," in Proc. of the IEEE GLOBECOM 2006, pp. 1-5, San Francisco, CA, USA, November-December 2006.
- [164] W. -K. K. Yiu, X. Jin, and S.-H.G. Chan, "Challenges and Approaches in Large-Scale P2P Media Streaming," IEEE Multimedia, Vol. 14, Issue 2, pp. 50-59, April 2007.
- [165] M. Hefeeda and C.-H. Hsu, "Rate-Distortion Optimized Streaming of Fine-Grained Scalable Video Sequences," in ACM Transactions on Multimedia Computing, Communications and Applications (TOMCCAP), Vol. 4, Num. 1, Article 2, pp. 1-28, January 2008.
- [166] ITU-T, "Video Codec for Audiovisual Services at 64 kbit/s," ITU-T Recommendation H.261, Version 1: November 1990; Version 2: Mar. 1993.
- [167] ITU-T, Video coding for low bit rate communication, ITUT Recommendation H.263; version 1, Nov. 1995; version 2, January 1998; version 3, November 2000.
- [168] ISO/IEC JTC 1, "Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mbit/s Part 2: Video," ISO/IEC 11172-2 (MPEG-1), March 1993.
- [169] ITU-T and ISO/IEC JTC 1, "Generic Coding of Moving Pictures and Associated Audio Information Part 2: Video," ITU-T Recommendation H.262 ISO/IEC 13818-2 (MPEG-2), November 1994 (with several subsequent amendments).
- [170] ISO/IEC JTC 1, "Coding of Audio-Visual Objects part 2: Visual," ISO/IEC 14496-2 (MPEG-4 Part 2), January 1999 (with several subsequent amendments).
- [171] T. Wiegand, G. J. Sullivan, G. Bjφntegaard, and A. Luthra, "Overview of the H.264/AVC Video Coding Standard," in IEEE Transaction on Circuit and Systems for Video Technology, Vol. 13, Num. 7, pp. 560-576, July 2003.
- [172] D. Marpe, T. Wiegand, and G. J. Sullivan, "The H.264/MPEG4 Advanced Video Coding Standard and its Applications," in IEEE Communications Magazine, pp. 134-142, August 2006.
- [173] "Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Recommendation H.264/ISO/IEC 14 496-10 AVC)," in Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, JVTG050, 2003.

- [174] G. J. Sullivan, P. Topiwala, and A. Luthra, "The H.264/AVC Advanced Video Coding Standard: Overview and Introduction to the Fidelity Range Extensions," in Proc. of the 27th SPIE Conference on Applications of Digital Image Processing, Special Session on Advanced in the New Emerging Standard: H.264/AVC, Vol. 5558 (2), pp. 454-474, Denver, CO, USA, August 2004.
- [175] I. Richardson, "An Overview of H.264 Advanced Video Coding," Vcodex white paper, 2007.
- [176] H. Schwarz, D. Marpe, and T. Wiegand, "MCTF and Scalability Extension of H.264/AVC," in Proc. of the Picture Coding Symposium (PCS), San Francisco, CA, USA, December 2004.
- [177] T. Wiegand, "Scalable Video model 3.0. Joint Video Team (JVT)," January 2005.
- [178] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the Scalable Video Coding Extension of the H.264/AVC Standard," in IEEE Transactions on Circuits and Systems for Video Technology, Vol. 17, No. 9, pp. 1103-1120, September 2007.
- [179] J. R. Ohm, "Advances in Scalable Video Coding," in IEEE, Vol.93, No.1, pp. 42-56, January 2005.
- [180] R. Schäfer, H. Schwarz, D. Marpe, T. Schierl, and T. Wiegand, "MCTF and Scalability Extension of H.264/AVC and its Application to Video Transmission, Storage, and Surveillance," in Proc. of the Visual Communications & Image Processing (VCIP'05), July 2005.
- [181] H. Schwarz, T. Hinz, D. Marpe, and T. Wiegand, "Constrained Inter-Layer Prediction for Single-Loop Decoding in Spatial Scalability," in Proc. of the IEEE International Conference on Image Processing, 2005. (ICIP'05), Vol. 2, pp. 870-873, Genova, Italy, September 2005.
- [182] Fraunhofer HHI homepage. http://ip.hhi.de/imagecom_G1/savce/downloads/SVC-Reference-Software.htm seen on 05.2008.
- [183] ITU-R.BT.601-5, "Studio Encoding Parameters of Digital Television for Standard 4:3 and Wide-screen 16:9 Aspect Ratios," 1995 (formelly CCIR-601).
- [184] S. Floyd and K. Fall, "Promotion the Use of End-to-end Congestion Control in the Internet," in IEEE/ACM Transaction Networking, Vol. 7, Num.4, pp. 458-472, August 1999.
- [185] J. Padhye et al., "Modeling TCP Reno Performance: A Simple Model and its Empirical Validation," in IEEE/ACM Transaction Networking, Vol. 8, Num.2, pp. 133-145, April 2000.
- [186] J. Widmer, R. Denda, and M. Mauve, "A Survey on TCP-Friendly Congestion Control," in IEEE Network, pp. 28-37, May/June 2001.
- [187] X. Li, M. H. Ammar, and S. Paul, "Video Multicast over the Internet," in IEEE Network, Vol. 13, March-April, pp.46-60, 1999.
- [188] M. Gondran and M. Minoux, "Graph and Algorithms," John Wiley & Sons Ltd., 1984.
- [189] R. Albert and A. L. Barabási, "Statistical Mechanics of Complex Networks," in Reviews of Modern Physics, Vol. 74, pp. 47-97, January 2002.
- [190] E. Ravasz and A. L. Barabási, "Hierarchical Organization in Complex Networks," in Physical Rewiev E67. 026112, 2003.

- [191] R. Xu and D. Wunsch, "Survey of Clustering Algorithm," IEEE Transactions on Neural Networks, Vol. 6, Num. 3, pp. 645-678, May 2005.
- [192] A. L. Barabási and E. Bonabeau, "Scale-free Networks," Scientific American, pp. 50-59, May 2003.
- [193] M. E. J. Newman, "The Structure and Function of Complex Networks," in SIAM Review, Vol. 45, Issue 2, pp. 167-256, 2003.
- [194] L. Mathy, R. Canonico, S. Simpson, and D. Hutchison, "Scalable Adaptive Hierarchical Clustering," in Proc. of the Second International IFIP-TC6 Networking Conference (NETWORKING'02), pp. 1172-1177, Pisa, Italy, May 2002.
- [195] S. Basagni, I. Chlamtac, and A. Farago, "A Generalized Clustering Algorithm for Peerto-Peer Networks," in Proc. of the Workshop on Algorithmic Aspects of Communication, Bologna, Italy, July 1997.
- [196] K. Hammouda and M. Kamel, "HP2PC: Scalable Hierarchically-Distributed Peer-to-Peer Clustering," in Proc. of the Seventh SIAM International Conference on Data Mining (SDM07), Minneapolis, MN, April 2007.
- [197] C. Diot, B. Levine, B. Lyles, H. Kassem, and D. Balensiefen, "Deployment Issues for the IP Multicast Service and Architecture," in IEEE Network, Vol. 14, Num. 1, pp. 78-88, January/February 2000.
- [198] A. Ganjam and H. Zhang, "Internet Multicast Video Delivery," in Proceedings of the IEEE, Vol. 93, Num. 1, pp. 159-170, January 2005.
- [199] L. Jia, A. Bagirov, I. Ouveysi, and A. M. Rubinov, "Optimization based Clustering Algorithms in Multicast Group Hierarchies," in Proc. of the Australian Telecommunications Networks and Applications Conference (ATNAC 2004), Melbourne Australia 2003.
- [200] J. B. MacQueen, "Some Methods for classification and Analysis of Multivariate Observations," in Proc. of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Vol. 1, pp. 281-298, Berkeley, CA, USA, December 1965-January 1966.
- [201] W. R. Stevens, B. Fenner, and A. M. Rudoff, "UNIX Network Programming," Vol. 1, 2003.
- [202] N. Wirth, "Algorithms and Data Structure," Prentice Hall, 1985.
- [203] A. V. Aho, J. Ullman, and J. E. Hopcroft, "Data structures and Algorithms," Addison Wesley, 1983.
- [204] W. L. Winston, "Operations Research: Applications and Algorithm," Duxbury Press, 1987.
- [205] D. R. Butenhof, "Programming with POSIX threads," Addison Wesley Longman, 1997.
- [206] M. Garey and D. S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness," W.H. Freeman and Company, 1979.
- [207] R. Ahlswede, N. Cai, S. R. Li, and R. W. Yeung, "Network Information Flow," in IEEE Transactions on Information Theory, Vol. 46, Num. 4, pp. 1204-1216, July 2000.
- [208] C. Gkantsidis and P. Rodriguez, "Network Coding for Large Scale Content Distribution," in Proc. of the IEEE INFOCOM'05, Vol. 4, pp. 2235-2245, Miami, FL, USA, March 2005.

- [209] P. A. Chou, Y. Wu, and K. Jain, "Practical Network Coding," in Proc. of the 41st Allerton Conference on Communications, Control and Computing, Urbana-Champaign, IL, USA, October 2003.
- [210] D. M. Chiu, R. W. Yeung, J. Huang, and B. Fan, "Can Network Coding Help in P2P Networks," in Proc. of the Second Workshop of Network Coding, in conjunction with WiOpt'06, Boston, MA, USA, April 2006.
- [211] K. Chen and Q. Deng, "Legitimate Peer-to-Peer Content Distribution Network," in Proc. of the Fifth International Conference on Grid and Cooperative Computing Workshops (GCCW'06), pp. 129-132, Changsha, Hunan, China, 21-23, October 2006.
- [212] T. S. E. Ng and H. Zhang, "Predicting Internet Network Distance with Coordinatesbased Approaches," in Proc. of the IEEE INFOCOM'02, Vol. 1, pp. 170-179, New York, NY, USA, June 2002.
- [213] Cooperative Association for Internet Data Analysis (CAIDA) homepage. http://www.caida.org. seen on 05.2008.
- [214] SID video cutter and splitter software homepage. http://www.soundindepth.com
- [215] J. Liu, S. G. Rao, and B. Li, "Opportunities and Challenges of Peer-to-Peer Internet Video Broadcast," in Proceedings of the IEEE, Vol. 96, Num. 1, pp. 1-24, January 2008.
- [216] S. Agarwal, "A Case Study of Large Scale P2P Video Multicast," in Proc. of the IEEE International Conference on IP Multimedia Subsystems Architecture and Applications (IMSAA07), Bangalore, India, December 2007.
- [217] S. Birrer and F. E. Bustamante, "Magellan: Performance-based, Cooperative Multicast," in Proc. of the 10th International Workshop on Web Content Caching and Distribution (WCW'05), pp. 133-143, Sophia Antipolis, France, September 2005.
- [218] Q. Zhang, Q. Guo, Q. Ni, W. Zhu, and Y. -Q. Zhang, "Sender-Adaptive and Receiver-Driven Layered Multicast for Scalable Video Over the Internet," in IEEE Transactions on Circuits and Systems for Video Technology, Vol. 15, Num. 4, April 2005.
- [219] C. Liang, Y. Guo, and Y. Liu, "Hierarchically Clustered P2P Streaming System," in Proc. of the IEEE GLOBECOM 2007, pp. 236-241, Washington, DC, USA, November 2007.
- [220] X. -B. Liang, "Matrix Games in the Multicast Networks: Maximum Information Flows With Network Switching," in IEEE Transactions on Information Theory, Vol. 52, Num.6, June 2006.
- [221] Y. Liu, Y. Guo, and C. Liang, "A Survey on Peer-to-Peer Video Streaming Systems," in Peer-to-Peer Network Applications, pp. 18-28, Vol. 1, Num. 1, March 2008.
- [222] M. Chen, M. Ponec, S. Sengupta, J. Li, and P. A. Chou, "Utility Maximization in Peerto-Peer Systems," in Proc. of the ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS 2008), pp. 169-180, Annapolis, MD, USA, June, 2008.