TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

Jani Peltotalo

Solutions for Large-Scale Content Delivery over the Internet Protocol

Julkaisu 925 • Publication 925

Tampere 2010

Tampereen teknillinen yliopisto. Julkaisu 925
Tampere University of Technology. Publication 925

Jani Peltotalo

Solutions for Large-Scale Content Delivery over the Internet Protocol

Thesis for the degree of Doctor of Science in Technology to be presented with due permission for public examination and criticism in Tietotalo Building, Auditorium TB109, at Tampere University of Technology, on the 19th of November 2010, at 12 noon.

Tampereen teknillinen yliopisto - Tampere University of Technology
Tampere 2010

ISBN 978-952-15-2457-8 (printed)
ISBN 978-952-15-2506-3 (PDF)
ISSN 1459-2045

ABSTRACT

The current trend is going more and more towards IP-based delivery for all kind of digital content. At the same time, the increasing quality of the digital media is simultaneously increasing the size of the media, which will also increase the requirements for the capacity in the content delivery path. It is obvious that the traditional content delivery based on the client-server model will easily overload the delivery network and none of the customers will be happy about the quality of the service. Hence, more scalable solutions in the digital content distribution area are needed.

In the best-effort service, like in the IP datagram forwarding, the successful delivery of a packet to its receivers is not guaranteed in the network layer. So, in IP-based applications, failures in the content delivery path between the sender and receiver will cause packet losses, which have to be dealt with in the transport or application layers. Another reason for packet losses in a multi-sender P2P environment is a peer churn which will cause the media being sent to a receiver to be temporarily interrupted.

This Thesis studies large-scale content delivery over the Internet Protocol, mainly at the *scalability* and *reliability* point of view, in two separate research areas: (a) file delivery to large user population, and (b) real-time P2P media streaming in a mobile networking environment. Multicast-based file delivery is one of the most efficient ways to deliver the same content to a large user population, but reliability becomes a concern, because multicast techniques are commonly based on unreliable transport protocols to scale up to large and massive receiver groups. As it is shown in this Thesis, FEC data carousel will be the best way to provide reliability in most cases when the total amount of data which is transmitted in the delivery system is used as a critical factor.

In P2P content distribution, overlay network structure and data partitioning are very important issues from the scalability point of view. A random mesh-based overlay architecture provides flexibility for handling peer departures, but good general con-

nectivity between peers is better achieved using for example clustered overlay architecture. In P2P delivery, a downloading client becomes a leecher peer when it has at least one complete block, so with a small enough block size the number of alternative source peers will increase faster. Data partitioning in P2P media streaming applications is even more demanding. Partitioning based on fixed byte ranges, like in P2P file delivery, is not suitable for streaming a continuous media, which is of variable bit rate nature.

In contrast to file delivery where one does not care if the data parts arrive in the original order or not, since the viewing experience will be anyhow the same once the file is fully downloaded, P2P media streaming applications require that all data is received relatively close to its playback position. Good user experience is achieved by using client side buffers to eliminate the network induced delay and jitter. With bigger buffer size it is possible to smooth the variation between packet arrival times and have also time for packet loss recovery. On the other hand, the smaller the buffering time is the faster the playback can be started. The real-time P2P media streaming system presented in this Thesis contains several important improvements to enhance the mobile usage, like small ten seconds initial buffering time, ten seconds reception buffer due to the RTP usage, and the partial RTP stream concept, which allow a single media stream to be effectively received simultaneously from multiple senders.

PREFACE

The work presented in this Thesis has been carried out at the Department of Communications Engineering at the Tampere University of Technology during the years 2003-2010. The research work has been funded by Nokia Research Center, TeliaSonera, Tekes – the Finnish Funding Agency for Technology and Innovation and Tampere University of Technology Doctoral Programme. In addition, I have received grants form Research and Training Foundation of TeliaSonera Finland Oyj and from Industrial Research Fund at the Tampere University of Technology. All aforementioned financial support is appreciatively acknowledged.

The work would not be possible without the help and support of many people. First of all, I would like to express my gratitude to my supervisor Prof. Jarmo Harju for his guidance and motivation during the research. Sincere acknowledgement goes also to the reviewers of the Thesis, Prof. Jörg Ott and Dr. Vincent Roca, for their valuable comments and criticism which helped me to improve the quality of the Thesis. The research work for all publications included in this Thesis has been done in a team environment, so I want to thank all people who have worked with me in several research projects during the years. In addition, I would like to thank all other colleagues in our department for the very pleasant working environment.

I would also like to thank my mother and late father for the continuous support and care. Finally, many thanks go to my dear wife Heidi for her love and support and to our beautiful children, Peppi, Veikka and Unna, for the joy they bring to my life.

Tampere, October 2010

Jani Peltotalo

TABLE OF CONTENTS

LIST OF PUBLICATIONS

This Thesis consists of an introductory part and the following publications that have been previously published. In the text, these publications are referred to as [P1], [P2], ..., [P7].

[P1] Jani Peltotalo, Sami Peltotalo, Jarmo Harju, and Rod Walsh, "Performance analysis of a file delivery system based on the FLUTE protocol," in *International Journal of Communication Systems*, Volume 20, Issue 6, October 5 2006, pp. 633–659. doi:10.1002/dac.835

[P2] Jani Peltotalo, Sami Peltotalo, Alex Jantunen, Lassi Väätämöinen, Jarmo Harju, Rami Lehtonen, and Rod Walsh, "A Massively Scalable Persistent Content Distribution System," in *Proceedings of the Sixth IASTED International Conference on Communications, Internet, and Information Technology (CIIT 2007)*, Banff, Alberta, Canada, July 2–4 2007, pp. 255–261.

[P3] Jani Peltotalo, Jarmo Harju, Alex Jantunen, Marko Saukko, Lassi Väätämöinen, Igor D. D. Curcio, Imed Bouazizi, and Miska M. Hannuksela, "Peer-to-Peer Streaming Technology Survey," in *Proceedings of the Seventh International Conference on Networking (ICN 2008)*, Cancun, Mexico, April 13–18 2008, pp. 342–350. doi:10.1109/ICN.2008.86

[P4] Jani Peltotalo, Jarmo Harju, Marko Saukko, Lassi Väätämöinen, Igor D. D. Curcio, and Imed Bouazizi, "Personal Mobile Broadcasting based on the 3GPP MBMS System," in *Proceedings of the 6th International Conference on Advances in Mobile Computing & Multimedia (MoMM2008)*, Linz, Austria, November 24–26 2008, pp. 156–162. doi:10.1145/1497185.1497219

[P5] Jani Peltotalo, Jarmo Harju, and Miska M. Hannuksela, "Reliable, Server-Friendly and Bandwidth-Efficient File Delivery System using FLUTE Server

File Format," in *Proceedings of the IEEE International Symposium on Broadband Multimedia Systems and Broadcasting 2009 (BMSB2009)*, Bilbao, Spain, May 13–15 2009, pp. 1–6. doi:10.1109/ISBMSB.2009.5133753

[P6] Jani Peltotalo, Jarmo Harju, Lassi Väätämöinen, Igor D. D. Curcio, and Imed Bouazizi, "RTSP-based Mobile Peer-to-Peer Streaming System," in *International Journal of Digital Multimedia Broadcasting*, Volume 2010, Article ID 470813, 15 pages, 2010. doi:10.1155/2010/470813

[P7] Jani Peltotalo, Jarmo Harju, Lassi Väätämöinen, Igor D. D. Curcio, Imed Bouazizi, and Joep van Gassel, "Scalable Packet Loss Recovery for Mobile P2P Streaming," in *Proceedings of the Eighth International Conference on Wired/Wireless Internet Communications (WWIC 2010)*, LNCS 6074, Luleå, Sweden, June 1–3 2010, pp. 107–120. doi:10.1007/978-3-642-13315-2_9

LIST OF FIGURES

LIST OF ABBREVIATIONS

3GPP	Third Generation Partnership Project
ADSL	Asymmetric Digital Subscriber Line
ALC	Asynchronous Layered Coding
ALM	Application Layer Multicast
ARQ	Automatic Repeat Request
ASM	Any-Source Multicast
AVC	Advanced Video Coding
BB	Building Block
BCL	Backup Cluster Leader
BM-SC	Broadcast Multicast Service Centre
CBT	Clustered BitTorrent
CC	Congestion Control
CDN	Content Delivery Network
CL	Cluster Leader
CSRC	Contributing Source
DCCP	Datagram Congestion Control Protocol
DNS	Domain Name System
DoS	Denial of Service

DRM	Digital Rights Management
DVB	Digital Video Broadcasting
DVB-H	DVB – Handheld
DVD	Digital Versatile Disc
EDGE	Enhanced Data rates for GSM Evolution
ESI	Encoding Symbol Identifier
ESP	Encapsulating Security Payload
FD	File Delivery
FDT	File Delivery Table
FEC	Forward Error Correction
FICIX	Finnish Communication and Internet Exchange association
FLUTE	File Delivery over Unidirectional Transport
HSDPA	High Speed Downlink Packet Access
HSUPA	High Speed Uplink Packet Access
HTTP	Hypertext Transfer Protocol
HTTPS	HTTP over TLS
IGMP	Internet Group Management Protocol
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPDC	IP Datacasting
IPTV	IP Television
IPv4	IP version 4
IPv6	IP version 6

ISO	International Organization for Standardization
ISP	Internet Service Provider
LAN	Local Area Network
LCT	Layered Coding Transport
LDPC	Low Density Parity Check
MAD-FCL	MAD FileCasting Library
MBMS	Multimedia Broadcast Multicast Service
MD5	Message Digest 5
MDC	Multiple Description Coding
MPEG	Moving Picture Experts Group
MP4	MPEG-4 Part 14
MP3	MPEG-1 Audio Layer 3
MTU	Maximum Transmission Unit
NACK	Negative Acknowledgement
NAT	Network Address Translation
OSI	Open System Interconnection
P2P	Peer-to-Peer
PGP	Pretty Good Privacy
PI	Protocol Instantiation
PKI	Public Key Infrastructure
PPSP	Peer-to-Peer Streaming Protocol
PTM	Point-to-Multipoint
PTP	Point-to-Point

QoE	Quality of Experience
QoS	Quality of Service
RMT	Reliable Multicast Transport
RTP	Real-time Transport Protocol
RTCP	RTP Control Protocol
RTSP	Real Time Streaming Protocol
RTT	Round Trip Time
SAP	Session Announcement Protocol
SBN	Source Block Number
SDP	Session Description Protocol
SDS	Session Discovery Server
SHA-1	Secure Hash Algorithm 1
SRTP	Secure RTP
SSM	Source-Specific Multicast
SVC	Scalable Video Coding
STUN	Session Traversal Utilities for NAT
TCP	Transmission Control Protocol
TOI	Transport Object Identifier
TSI	Transport Session Identifier
TURN	Traversal Using Relays around NAT
UDP	User Datagram Protocol
UMTS	Universal Mobile Telecommunications System
URI	Uniform Resource Identifier

URL	Uniform Resource Locator
VoD	Video-on-Demand
VoIP	Voice over IP
VPN	Virtual Private Network
WEBRC	Wave and Equation Based Rate Control
WLAN	Wireless LAN
WWW	World Wide Web

1. INTRODUCTION

As the amount of media delivered in the Internet Protocol (IP) [104] based networks seems to be ever-growing and the speeds of end-users' access network connections are getting faster day by day, the core network capacity continues to be a scarce resource. Proxy servers [66, pp. 112-116] and Content Delivery Networks (CDNs) [21], such as Akamai [11], have been widely utilized to reduce the response time for a client request and the amount of data delivered in the core network when the client-server model is used in the content delivery. However, due to the client-server model a particular proxy server or CDN server might also become the bottleneck, like the original server without proxy server or CDN usage, if a massive number of clients located in the same area are requesting the content simultaneously, and most of those are using the same proxy server or are directed to the same CDN server. Coupled with the problems related to security, heterogeneity of the networks and rights management issues advanced solutions in the digital content distribution space are needed.

IP multicast was designed globally accessible, but the stateful method for routing packets between operators was considered too resource consuming and therefore global IP multicast distribution has not been much deployed. IP Television (IPTV) deployments within operators, like [85], [122], and [33] in Finland, have been one of the major IP multicast deployments so far and in those cases IP multicast has been enabled only internally and even just for the IPTV service usage. Those IP multicast enabled isolated networks are coming commonly available and can be also used for other than IPTV service. However, it is evident that IP multicast is not solution to all problems that exist in the large-scale content delivery. Nor does Peer-to-Peer (P2P) alone solve the distribution problem. Of course the bottleneck is no more in single links close to media sources, but is distributed more evenly in the network. Therefore, there is still demand for one-to-many delivery for static media files that can be distributed to multiple receivers at the same time. Thus, IP multicast has

great advantages for delivery with controlled last-mile elements, both for mobile and fixed usage. However, content persistence while the number of still-receiving users dwindle is better served by P2P techniques.

At the moment, P2P media streaming is securing its position among users. Although the improvements compared with a traditional unicast media transfer are great and will help small organizations to build, for example Internet radio service to a large audience rather easily, people are probably unaware of the opportunities. For the average user, YouTube [139] allows easy to access and easy to use services in order to spread user generated content and to share experiences in a community like environment. Nevertheless, if the intention is a professional broadcasting of high-quality content over the Internet to a large-scale audience, then P2P media streaming is a much better choice.

1.1 Objective and Scope of Research

The objective of this Thesis is to create enabling tools for content providers, service providers, network operators and end users to help the distribution and use of all kind of digital content. The target is to show that by combining different technologies it is possible to provide scalable and reliable file delivery to a massive user population. Additionally, this Thesis focuses on real-time P2P media streaming in a mobile networking environment. The increasing use of mobile devices in our daily life will also create demand for the new types of services in the mobile domain. Some of the currently existing P2P media streaming applications, such as Octoshape [92] and SopCast [123], are somehow suitable for being used in a mobile networking environment but still more tailored applications where the whole concept is designed with a mobile environment philosophy are needed to gain support from mobile users.

The scope of this Thesis is large-scale content delivery over the Internet Protocol. Basically this Thesis combines prototype solutions for multicast-based reliable file delivery system and P2P-based real-time media streaming system. This Thesis is mainly focusing on the challenges and solutions in large-scale content delivery and is targeting people with information and communication technology backgrounds.

1.2 Main Contributions

To summarize, the main contributions of the Thesis are:

- Implementation and performance analyses of a reliable, server-friendly and bandwidth-efficient file delivery system
- A brief survey on the P2P media streaming field and a closer look at selected applications
- Implementation and performance analyses of a large-scale real-time P2P media streaming system for a mobile networking environment
- The specification of a personal mobile broadcasting system as an extension to the currently existing Third Generation Partnership Project (3GPP)[1] Multimedia Broadcast Multicast Service (MBMS) system

1.3 Author's Contribution

The research work for all publications included in this Thesis has been done in a team environment where all of the authors have contributed to the work. The first author of the publication is however identified as a main contributor. The author of this Thesis has been the main contributor in all of the included publications. As a part of the research work, the author of this Thesis has also contributed to standardization activities in the Internet Engineering Task Force (IETF)[2], in the 3GPP and in the Moving Picture Experts Group (MPEG)[3].

Publications can be divided into two main groups. Publications [P1], [P2], and [P5] propose components for a reliable large-scale file delivery system. Publications [P3], [P6], and [P7] deal with a P2P media streaming in a mobile networking environment. Publication [P4] is a little side track, by proposing personal mobile broadcasting still utilizing the same content delivery protocols as used in the other publications.

[1] http://www.3gpp.org/
[2] http://www.ietf.org/
[3] http://mpeg.chiariglione.org/

Publication [P1] presents the results of performance tests carried out for a file delivery system based on the File Delivery over Unidirectional Transport (FLUTE) [97, 98] protocol. The publication shows how FLUTE manages to recover from packet losses using Forward Error Correction (FEC) and repeat transmissions in a data carousel, and a simple Point-to-Point (PTP) file repair scheme based on the Hypertext Transfer Protocol (HTTP) [38]. The results show that careful optimisation of FEC overhead, and the number of repeat transmissions, gives the best system performance in most cases. Based on the simplified error reception and distribution model, it is mathematically illustrated that the simple client-server PTP file repair is optimal only for small groups.

The author has been the main contributor for the MAD FileCasting Library (MAD-FCL) library [83] used in the performance tests. The initial version of the publication was written together with M.Sc. Sami Peltotalo, and all modifications after the peer review were done by the author. M.Eng Rod Walsh revised the text in the final phase and provided some additional input to the publication. Prof. Jarmo Harju improved the writing style.

Publication [P2] proposes a large-scale content distribution system based on IP multicast and P2P delivery technique, with a need for timely and reliable delivery, and content persistence. The strengths of both IP multicast and P2P overlay have been leveraged to simultaneously scale server and distribution network capacities to serving a greater number of receiving hosts for mass media content.

The author provided necessary changes for the MAD-FCL library together with M.Sc. Sami Peltotalo as a continuation for the implementation work done for [P1]. The client side of the Delco content delivery system [30] was implemented by Mr. Alex Jantunen, and the server side by Mr. Lassi Väätämöinen. The text was mainly written by the author, but all other authors provided also some input to the publication. M.Sc. Rami Lehtonen, M.Eng Rod Walsh and Prof. Jarmo Harju contributed to the system designing and revised the text.

Publication [P3] gives a brief survey on the P2P media streaming field and takes also a closer look at selected applications. In practice, selected P2P media streaming systems, Octoshape, SopCast, TVAnts [127] and TVU networks [128], are analysed and tested over different leased line and mobile network connections. Experimental tests showed that all applications are somehow suitable for mobile usage with high-

throughput mobile networks, but still it is clear that they are designed to be used with leased line connections.

Performance measurements for the publication were done by the author and Mr. Marko Saukko. The text was mainly written by the author. Some parts of the publication contain rewritten text provided by M.Sc. Alex Jantunen, Mr. Lassi Väätämöinen and Mr. Marko Saukko. Dr. Imed Boazizi, M.Sc. Igor D.D. Curcio, and M.Sc. Miska M. Hannuksela provided some preliminary material about the studied systems. M.Sc. Igor D.D. Curcio and Prof. Jarmo Harju revised the text and improved the writing style.

Publication [P4] presents a personal mobile broadcasting system as an extension to the currently existing 3GPP MBMS system. A new user-level interface between the Broadcast Multicast Service Centre (BM-SC) and the personal mobile broadcaster is proposed to provide the enablers for announcing, setting up and tearing down a personal mobile broadcast session and delivering media data to the MBMS user population. In addition some new functionality, to be able to restrict the delivery only to a subset of the MBMS user population, are proposed in the publication.

The text was written by the author based on preliminary material provided by M.Sc. Lassi Väätämöinen and Mr. Marko Saukko. Dr. Imed Boazizi and M.Sc. Igor D.D. Curcio gave guidance during the system designing phase. Prof. Jarmo Harju revised the text and improved the writing style.

Publication [P5] presents a reliable, server-friendly and bandwidth-efficient file delivery system using the FLUTE server file format. As is shown in [P1], the use of FEC is a good option to improve the reliability from the viewpoints of receivers and the file delivery system. However, the load of the file delivery server might increase if the FEC encoding is done on-the-fly. The FLUTE server file format enables storage of pre-composed source symbols and pre-calculated FEC symbols into a media container file, so there is no need to source symbol construction and FEC encoding on-the-fly.

The author implemented needed modifications to the MAD-FCL library and to the proprietary Raptor FEC [76] and MPEG-4 Part 14 (MP4) file format [53] libraries, and carried out the performance measurements. The publication was mainly written by the author of this Thesis. M.Sc. Miska M. Hannuksela provided some preliminary text and revised the text. Prof. Jarmo Harju improved the writing style.

Publication [P6] proposes an effective real-time P2P media streaming system for the mobile networking environment as an alternative solution to traditional streaming applications based on the client-server approach. A scalable overlay network which groups peers into clusters according to their proximity is created and maintained using extended Real Time Streaming Protocol (RTSP) [117] messages by the cluster leaders with the help of a service discovery server. Furthermore, the actual media delivery is implemented using a partial RTP stream concept. Real-time Transport Protocol (RTP) [116] sessions are split into a number of partial streams in such a way that it allows re-assembling the original media session in real-time at the receiving end. Partial streams also help in utilizing the upload capacity with finer granularity than just per one original stream. This is beneficial in mobile environments where bandwidth can be scarce.

The author has been member of the implementation team together with M.Sc. Lassi Väätämöinen, M.Sc. Alex Jantunen, M.Sc. Joep van Gassel, and Mr. Marko Saukko. The text has been written by the author based on discussions and preliminary material provided by the whole implementation team. Experimental tests presented in the publication have been conducted by the author and M.Sc. Lassi Väätämöinen. Dr. Imed Boazizi, M.Sc. Igor D.D. Curcio, and Prof. Jarmo Harju have been involved in the system designing and revised the text.

Publication [P7] presents a scalable two-stage packet loss recovery mechanism for a real-time P2P media streaming system using RTP Control Protocol (RTCP) [116] and RTSP. This publication is a direct continuation for [P6] to ensure seamless media playback, if some data packets are lost due to the sudden uncontrolled disappearance of a sender. Scalability is necessary to prevent the ripple through effect of retransmission requests and redundant retransmissions in case many peers start simultaneously requesting lost packets due to a single cause, like a failing or departed peer.

The author identified problems relating to RTCP-based packet loss recovery in a P2P environment. Scalable two-stage packet loss recovery mechanism is designed based on discussions in our project meetings. The author implemented the currently existing packet loss recovery mechanism and conducted experimental tests for the publication. The text is mainly written by the author based on preliminary material produced in the designing phase. M.Sc. Igor D.D. Curcio, and Prof. Jarmo Harju revised the text and improved the writing style.

1.4 Thesis Outline

This Thesis consists of an introductory part and seven publications [P1-P7]. The introductory part gives technical background information and describes what kind of challenges exist in a large-scale content delivery. Several ways how to enhance scalability and guarantee reliability are also highlighted in the introductory part. The main results are presented in the publications.

The rest of the introductory part is organized as follows. Chapter 2 introduces technologies that can be used in a large-scale content delivery over the IP. The chapter is divided into two part; first multicast and broadcast technologies are briefly covered, followed by an introduction to the P2P technologies. Chapter 3 focuses on the challenges when designing a large-scale content delivery system. The chapter covers issues relating to problems and obstacles for an application developer, service provider and end user which have to be taken into account in order to gain massive public support for the content delivery system.

Chapter 4 introduces methods how to provide reliability in a large-scale content delivery. The chapter is divided into two parts; first methods for guaranteeing the reliability in multicast-based file delivery applications are presented, followed by ways to ensure good user experience in P2P media streaming. Chapter 5 concentrates on the ways how to enhance system wide scalability in multicast-based file delivery applications and P2P media streaming applications. Finally, Chapter 6 concludes the Thesis.

2. TECHNOLOGIES FOR LARGE-SCALE CONTENT DELIVERY

In the Internet, IP is the protocol on which all digital contents are transmitted. Since IP is connectionless, it can be used in the unidirectional broadcast networks as well. Examples of a unidirectional system are MBMS system specified by the 3GPP and IP Datacast (IPDC) service[1] for Digital Video Broadcasting – Handheld (DVB-H) devices[2]. Considering Internet and unidirectional broadcast networks, IP acts as a unifying layer, giving possibilities for the combined use of these technologies.

Next, several key technologies, specified by the IETF, 3GPP, DVB project[3] and individual researchers, that can be used in a large-scale content delivery are briefly introduced.

2.1 Multicast and Broadcast Technologies

Multicast and broadcast are the most efficient ways to deliver the same content to a large user population. When serving multiple users with the same data using unicast delivery, the data is replicated at the server end separately for each user. This means that the same data packet is forwarded multiple times on each link depending on the amount of users behind that link. With a multicast and broadcast delivery, the amount of data delivered within the network can be remarkably reduced in comparison with the unicast delivery, since the same data packet is forwarded only once on each link for all users behind that link.

[1] http://www.ipdc-forum.org/

[2] http://www.dvb-h.org/

[3] http://www.dvb.org/

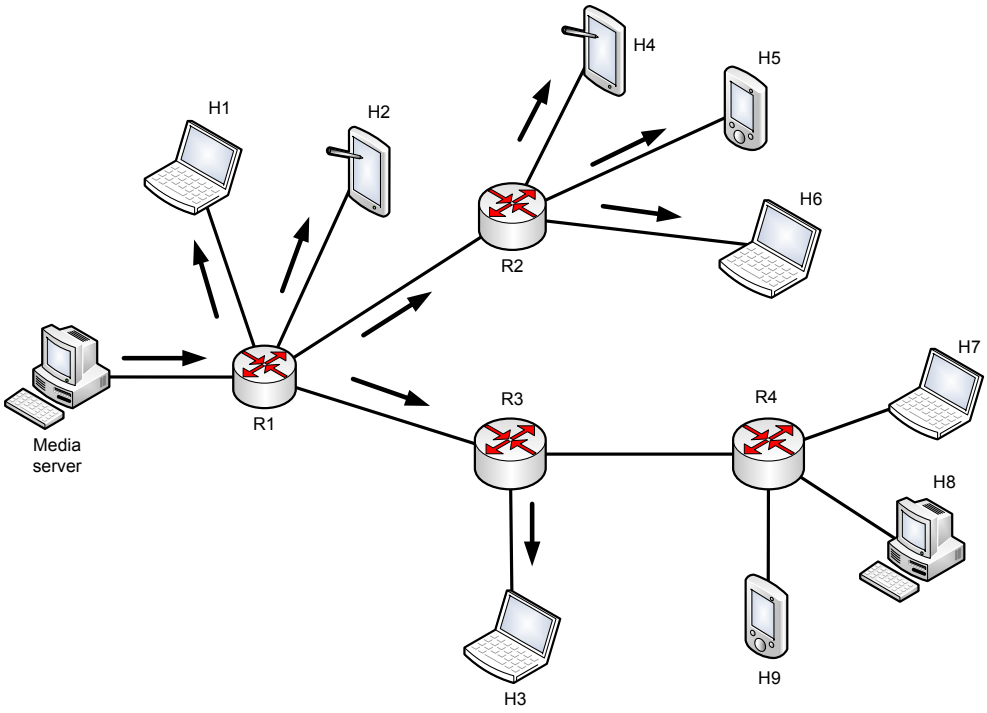

Fig. 1. *An example IP multicast delivery network*

2.1.1 IP Multicast

When using IP multicast, an IP datagram does not need to be replicated at the server end, but the same datagram is delivered for each receiver with a minimum amount of replication. This is achieved by using Point-to-Multipoint (PTM) delivery illustrated in Fig. 1. The receivers must join the multicast group using Internet Group Management Protocol (IGMP) [22] in order to receive the data delivered by the media server. The network infrastructure must also support multicast delivery to make it possible to forward the data stream to the receivers. If some of the routers do not support multicast delivery, like router *R4* in Fig. 1, the receivers located in the sub-network behind the router are not capable of receiving the data, even if desired.

There are two kinds of multicast service models. In Any-Source Multicast (ASM) [29], a receiver simply joins the multicast group and does not need to know the identity of the source(s). So, any host or router, whether or not belonging to the multicast group, may transmit data to all receivers belonging to that group. That is, an ASM multicast group may have an arbitrary number of transmitters delivering data to that

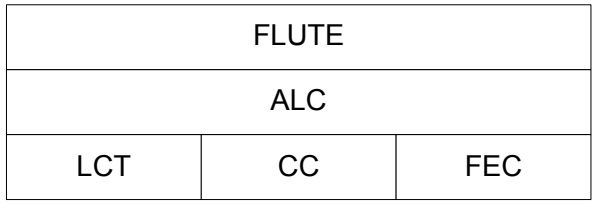

Fig. 2. *FLUTE BB structure*

group, hence the name any-source. Source-Specific Multicast (SSM) [47] modifies the ASM service model such that in addition to knowing the multicast group address, a receiver must also know relevant source(s). More information about IP multicast can be found in [134] and [32].

2.1.2 File Delivery over Unidirectional Transport

FLUTE is a protocol used to deliver files over IP networks, including the Internet and unidirectional systems, from a sender to one or more receivers. FLUTE can be used with both multicast and unicast User Datagram Protocol (UDP) [103] delivery, but it is particularly valuable in multicast networks.

FLUTE builds on the Asynchronous Layered Coding (ALC) Protocol Instantiation (PI) [80] of the Layered Coding Transport (LCT) Building Block (BB) [79]. LCT provides transport level support for reliable content delivery and stream delivery protocols. ALC combines the LCT BB, an optional Congestion Control (CC) BB (for example Wave and Equation Based Rate Control (WEBRC) Building Block [75]) and a FEC BB [133] to provide congestion controlled reliable asynchronous delivery. See Fig. 2 for an illustration of the FLUTE BB structure. This modular approach, taken by the Reliable Multicast Transport (RMT) IETF working group[4], allows to reuse existing components in different contexts and to enable or disable features according to the need. ALC PI is for example originally designed for the delivery of arbitrary binary objects and this service is used to deliver files with FLUTE.

LCT defines the notion of LCT channels to allow massive scalability, which has been designed based on the receiver-driven layered multicast principle, where receivers are responsible for implementing an appropriate CC algorithm based on the adding and removing of layers of the delivered data. A FLUTE session consists of one or

[4] http://datatracker.ietf.org/wg/rmt/charter/

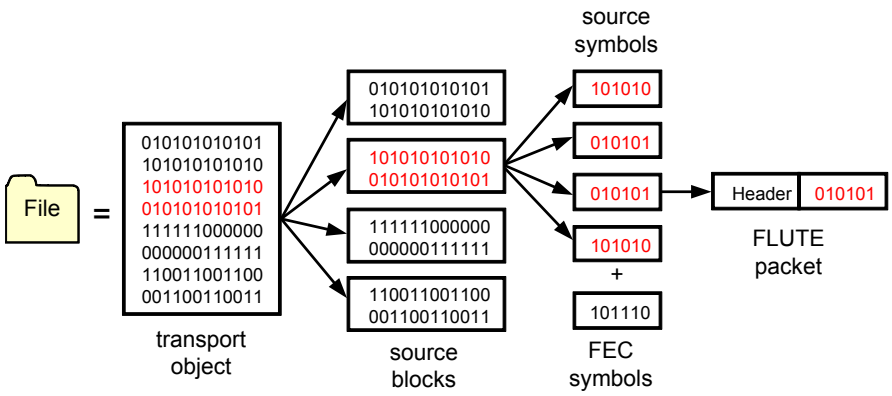

Fig. 3. *Building up a FLUTE packet*

more LCT channels identified by the combination of the sender (using the sender's IP address and a Transport Session Identifier (TSI) it assigns) and the multicast group (the IP multicast destination address).

Great flexibility is given to the FLUTE sender with regard to how the data is partitioned among the LCT channels. A common use case is to send the same content on all different LCT channels but encoded at different bit rates. Additionally, the FLUTE sender may act intelligently to enable receivers to acquire all files of the FLUTE session by joining all channels for a shorter time than is normally required with one channel. In such a case, the data sent over each channel complements the data of other channels.

The default FEC scheme for the FLUTE protocol is Compact No-Code FEC scheme [132], which provides file partitioning mechanism, i.e., how the file is divided into source blocks and source symbols, but does not produce FEC symbols, and so encoding symbols are all source symbols. However, when using, for example, Reed-Solomon FEC [68] or Raptor FEC schemes, FEC symbols are calculated and included in the encoding symbols.

Fig. 3 shows how a file is partitioned into FLUTE packets. The file is a transport object for the FLUTE protocol, and based on the file partitioning parameters of the used FEC scheme, a FLUTE sender calculates the source block structure, i.e., the number of source blocks and their lengths. Each source block is then fragmented into source symbols also according to the file partitioning parameters of the used FEC scheme. If FEC is used, then a desired number of FEC symbols are generated by the FEC

encoder based on the source symbols. Finally, a FLUTE packet is constructed with a FLUTE header and an encoding symbol and then it is ready for UDP/IP delivery.

The FLUTE sender communicates the metadata associated with the files delivered in the FLUTE session to the receivers using a special object, named File Delivery Table (FDT) instance. An FDT instance will be delivered with a dedicated Transport Object Identifier (TOI) value 0 and it describes various attributes, including those used in the file partitioning and thus the FLUTE receivers are able to calculate the source block structure in advance of receiving a file. An example FDT instance is shown in Listing 1.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<FDT-Instance Expires="3467015421">
  <File TOI="1"
    Content-Location="file:///flute"
    Content-Length="2229112"
    Content-MD5="bv0kXKlM300QtLqtggAAOw=="
    FEC-OTI-FEC-Encoding-ID="0"
    FEC-OTI-Maximum-Source-Block-Length="64"
    FEC-OTI-Encoding-Symbol-Length="1428"/>
</FDT-Instance>
```

Listing 1. *An example FDT instance*

In this example one file, named flute, will be delivered with TOI value 1 using Compact No-Code FEC scheme (FEC Encoding ID 0). The Message Digest 5 (MD5) [110] checksum allows receivers to check the integrity of the file after all source blocks are received. Other parameters (file size, maximum source block length and encoding symbol length) will be used by the partitioning algorithm to calculate the source block structure.

2.1.3 Real-time Transport Protocol

RTP provides end-to-end transport services for delivering real-time data over any kind of IP network. The majority of RTP implementations are using multicast or unicast UDP delivery, but RTP can be used also with other suitable underlying network or transport protocols, such as Datagram Congestion Control Protocol (DCCP) [65, 100]. Distribution of a real-time data simultaneously to many receivers can be

efficiently accomplished using PTM delivery, hence RTP is particularly valuable in multicast and broadcast networks where native support for PTM delivery is available.

The sending side precedes each data chunk with an RTP header to comprise an RTP packet. Information provided by the RTP header includes timestamps for jitter removal and synchronous rendering, sequence numbers for packet loss detection and concealment, and the payload format of the particular data chunk. The data transport is augmented by a control protocol (RTCP) to monitor the Quality of Service (QoS) and to provide minimal control and identification functionality.

In a multimedia session, each medium (such as audio, video or subtitle stream) is typically carried in a separate RTP session with its own RTCP packets unless the encoding itself bundles multiple media into a single stream during the encoding process. Each RTP session uses different pairs of destination transport addresses, where a pair of transport addresses consists of a network address plus a pair of ports for RTP and RTCP.

Base RTP specification [116] specifies only functions that are expected to be common across all RTP applications. Hence, a complete specification of RTP for a particular application will require one of more companion documents. An RTP profile specification, like [95, 115], will define a set of payload type codes and their mapping to payload formats, and an RTP payload format specification, such as [109, 115], will define how a particular payload is to be carried within RTP.

2.1.4 IP Datacast over DVB-H

IPDC over DVB-H [35] is a service for delivering all kind of digital content on top of IP over a unidirectional broadcast network. Because of the one-to-many broadcast delivery in the last-mile connection, it is an efficient way to reach large user population. An important aspect of the IPDC system is that it combines the unidirectional DVB-H [34] broadcast network and the bi-directional cellular mobile networks into a single overall service system. In practise this means that the DVB-H network is used for content delivery, and the mobile network is used for interaction. More information about DVB-H is available in [37].

Content delivery protocols for streaming and file delivery services are defined in [36]. RTP will be used for streaming services, where audio, video and subtitle streams

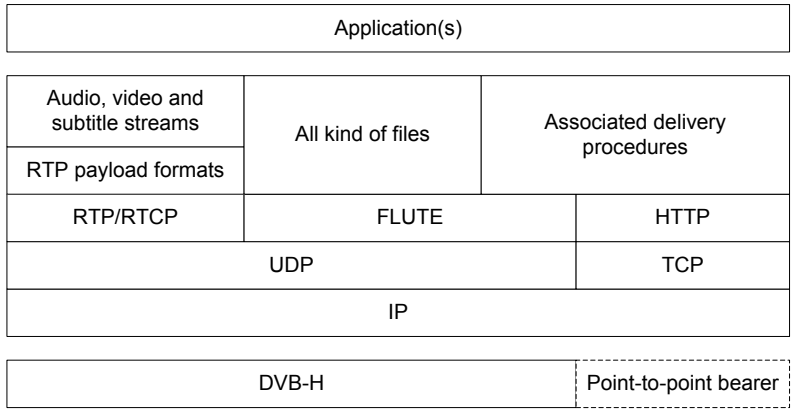

Fig. 4. *Simplified protocol stack for content delivery in IPDC over DVB-H*

are delivered in real-time and rendered while downloaded. FLUTE protocol will be used for non-real-time file delivery services, where files are first downloaded before consumption. Two associated delivery procedures are also applicable to the content delivery: post repair of files which are initially delivered as part of a FLUTE session and reception reporting allowing a receiver to report statistics about streaming or file delivery services. Simplified protocol stack for content delivery in IPDC over DVB-H is illustrated in Fig. 4.

2.1.5 Multimedia Broadcast Multicast Service System

MBMS is a PTM service where digital content is transmitted from a commercial content provider to a large number of users. MBMS service can de divided into two parts: MBMS bearer service [5] and MBMS user service [6]. MBMS bearer service defines physical broadcast and multicast channels with corresponding control signalling, and methods for the IP data delivery. MBMS user services are built over the MBMS bearer service and currently two different delivery methods, streaming and download, are defined.

Similar to the IPDC over DVB-H, RTP is used as a transport protocol for the streaming delivery method and FLUTE for the download delivery method. MBMS defines also the same two associated delivery procedures like IPDC over DVB-H, i.e., post repair of files and reception reporting. Simplified protocol stack for the MBMS user services is illustrated in Fig. 5. As it can be seen from Figs. 4 and 5, content delivery protocols are harmonized in the latest IPDC over DVB-H and MBMS specifications.

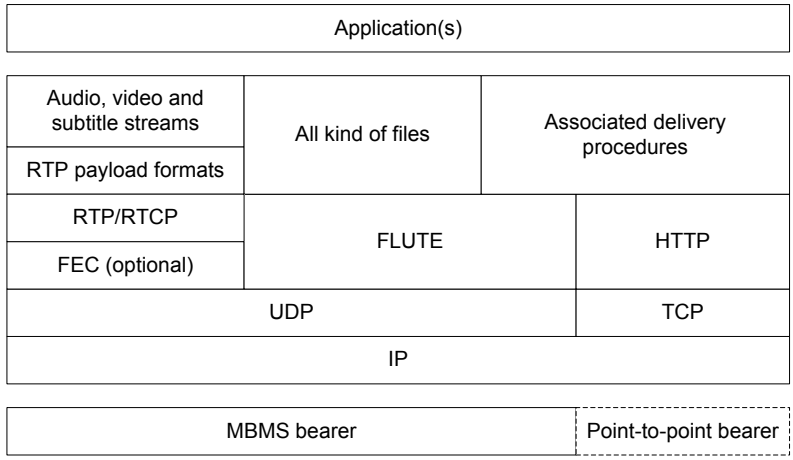

Fig. 5. *Simplified protocol stack for the MBMS user services*

The main difference is an optional FEC usage in the streaming delivery method in the latter one. Another observation from the figures is that both IPDC over DVB-H and MBMS rely heavily on the outcomes of the IETF working groups.

Almost identical protocol stacks allow service providers to utilize both technologies in the same service quite easily. This also increases the amount of possible users since the combined coverage area is usually bigger than individual coverage area. More information about content networking in the mobile Internet can be retrieved from [31, pp. 327-370], where also MBMS and IPDC systems are explained in detail.

2.2 Peer-to-Peer Technologies

In a P2P network each participant, a peer, shares some of its resources and can use resources offered by the other peers for its own purposes. Therefore, a single peer acts as a server and a client simultaneously. P2P networks can be classified in many ways. Usually, the classification is done based on the use case and P2P networks are divided into six distinct categories: file sharing, instant messaging, Voice over IP (VoIP) [41], storage networks, distributed computing, and media streaming.

Before being able to transfer any data in a P2P network, a peer must first somehow join the overlay network, and the means to do that vary between different P2P networks and protocols. Actual data delivery in a P2P network uses Application Layer Multicast (ALM) principle [48], where IP datagrams are replicated at the end

hosts, compared with native multicast delivery where IP datagrams are replicated at the routers. Since identical packets might be sent over the same link, depending on the overlay network structure, ALM is less efficient in comparison with the native multicast protocols. In contrast there is no need to change routers, so the network infrastructure does not restrict the service availability.

Although P2P techniques do not directly reduce the overall network load in comparison with the traditional client-server approach, the network load is distributed more evenly to the whole network. This leads to single links near to the media source being less congested and decreases also the total distribution time [66, pp. 147-151] since all peers are also redistributors as well as consumers of the data. More high-level information, like the definition of a P2P system, functions in a P2P system and taxonomies for P2P system can be fetch from [24]. The first textbook-like survey [125] provides also an up-to-date and in-depth introduction to the P2P field.

2.2.1 Peer-to-Peer Overlay Network Structures

A P2P network is actually an overlay network built on top of an existing network infrastructure. This means that the logical connections between peers are formed at a higher level than the network level typically using Transmission Control Protocol (TCP) [105]. In the overlay network the connections can be described as tunnels on the top of existing network topology that do not care about the underlying path.

Fig. 6 presents a tree layout for a P2P overlay network. Each peer in the network, except the data source, is connected to a parent node and several child nodes. In a tree-based overlay network a single point of failure type of problem exists. If any of the parent nodes happen to fail, the whole network originating from the failed parent node fails to receive the data. Even though tree is a very simple structure, it must have some maintenance functionalities, because peers are coming and going once in a while. Assume that *Peer 3* leaves, which simultaneously interrupts the data transfer that was going to its descendent, i.e., to *Peer 8*, *Peer 9*, *Peer 13* and *Peer 14*. The replacement for a missing data source must be found as fast as possible to prevent peers experiencing long interrupts in the data delivery. This is done by different search algorithms depending on the implementation.

The single point of failure issues can be avoided, or at least reduced in a combined tree-mesh or a mesh layout network. Fig. 7 illustrates mesh layout for a P2P over-

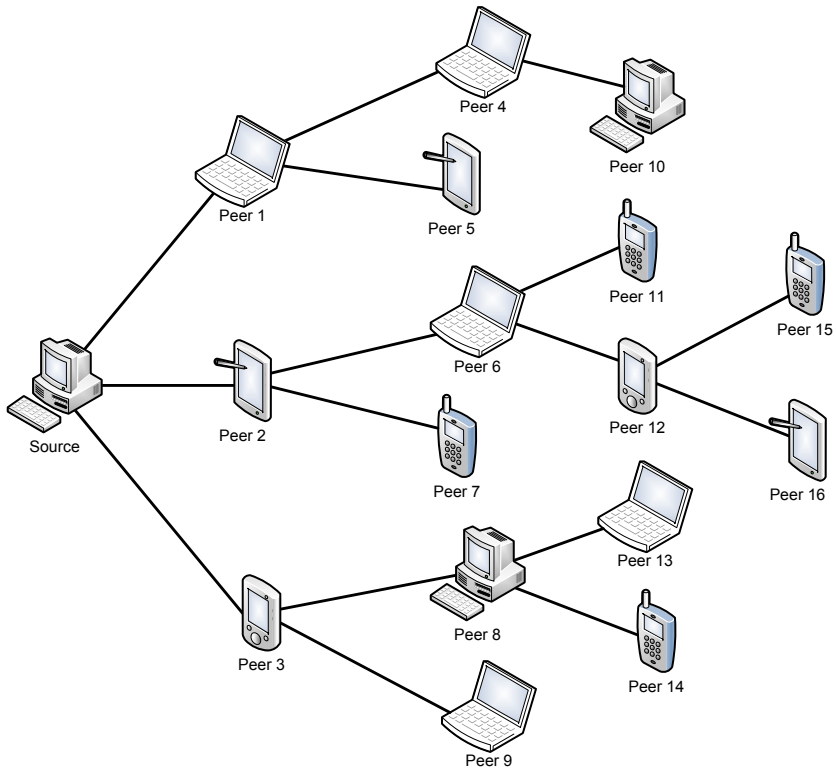

Fig. 6. *Tree-based P2P overlay network*

lay network where each peer is connected to more than one other peers. When a peer leaves the network, it only affects those peers that are connected to it. However, mesh layout requires more complicated routing algorithms or request mechanisms between peers due to the increased number of peer connections. A mesh-based overlay network is more commonly used in P2P file sharing than in P2P media streaming when real-time demands are not so important.

2.2.2 Peer-to-Peer File Sharing

Most of the P2P file sharing applications make use of multiple sources for information distribution. A file is first partitioned into pieces or chunks, typically of equal size. A peer then connects to the seeder or leecher peers and requests the missing pieces of the file in a random order. The difference between a seeder and a leecher peer is that the former has a complete copy of the file while the latter has only a partial copy and is sharing already downloaded pieces with the other peers while

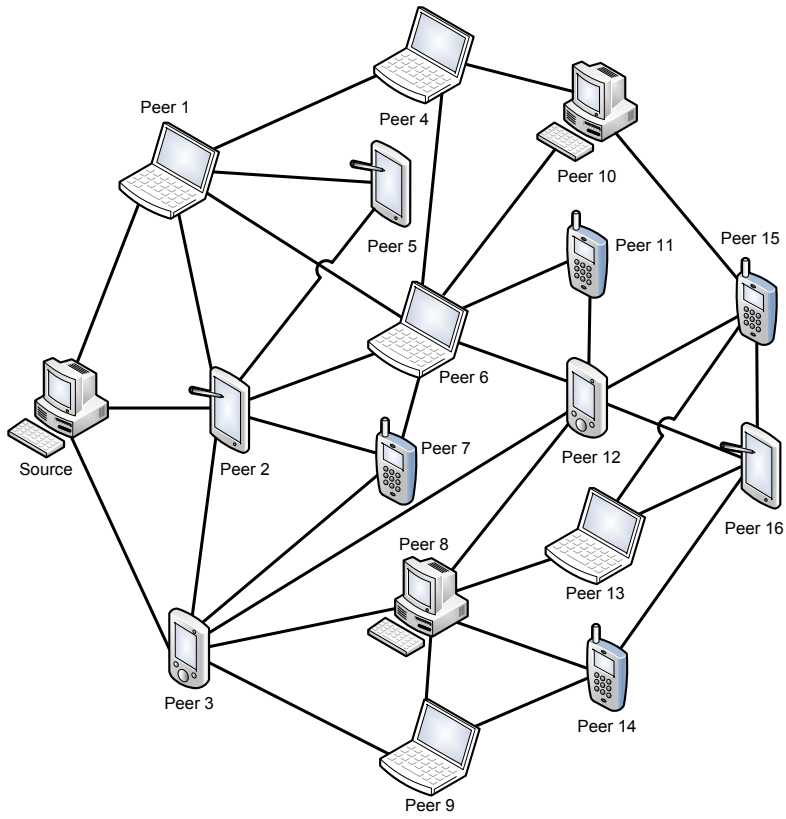

Fig. 7. *Mesh-based P2P overlay network*

downloading missing pieces at the same time.

At the time of writing this Thesis, one P2P file sharing network, BitTorrent [27] has almost gained de facto standard status when speaking about P2P file sharing. The BitTorrent overlay network consists of a tracker and seeder and leecher peers. Before a peer can join the overlay network a .torrent file must be first fetched somewhere, usually from some World Wide Web (WWW) server, and if the .torrent file is not available, all new downloads will be blocked. Search functionality is not part of the BitTorrent protocol, but often WWW servers with downloadable .torrent files are thought to be part of the BitTorrent system.

The file downloading process with BitTorrent is depicted in Fig. 8. The .torrent file retrieved in phase 1 contains a Uniform Resource Locator (URL) [19] of the tracker for the file sharing session in question. When a connection request from a peer is received, the tracker responds with a random list of other peers (phase 2), and these

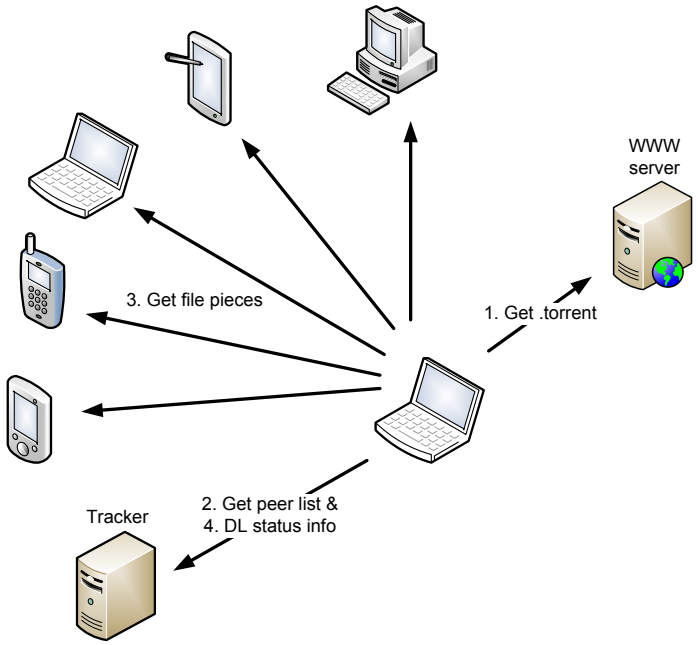

Fig. 8. *The file downloading process with BitTorrent*

randomly selected peers among all peers in the network might create a long delay to the file sharing between two peers. Peers request file pieces (phase 3) from several peers according to a rarest-first piece selection algorithm and after each file piece is correctly received a download status update message is sent to the tracker (phase 4). More information about the BitTorrent protocol can be found in [26]. BitTorrent has proven its strength for deploying large-scale P2P file delivery [59, 106], but there are still scalability issues with the original approach.

2.2.3 Peer-to-Peer Media Streaming

P2P media streaming is a method for multicasting or broadcasting streaming media over the Internet using a P2P network. It can be seen as a combination of traditional television or radio broadcast type of media delivery over a new kind of delivery medium. With real-time P2P media streaming there is no need to download the entire media file before playback can be started; decoding can be started as soon as enough data is buffered in the peer. The length of the buffering time depends on the amount of peers attending in the network, as well as the peers' network capacity and

the overall network latency. Also, both the software used for receiving and rendering the stream and the stream encoding format have an influence on the duration of the buffering time. In the live streaming, the video of an ongoing event, like a football match, is delivered as a stream in real-time. After an initial buffering period, media rendering is started from a certain location and all peers consume the data in the same time window. With a Video-on-Demand (VoD) streaming the user searches for a video from some catalogue, and after a certain amount of initial buffering playback is started from the beginning of the video.

P2P media streaming systems can be either push-based or pull-based [74, 141]. In a push-based system, a peer sends the data to other peers without explicit request after it has received the data. One of the problems with this approach is that the packet forwarding decision must be based on a predetermined routing algorithm, which makes the system somewhat inflexible. Loss recovery is also a big problem with push-based system. There is no request for the data and it is not possible to request retransmission of the missing data. Third problem is the amount of duplicate data because peers just forward the data based on the predetermined routing algorithm and this may lead to a situation that one or more peers will be sending the same packets to a common destination peer.

All aforementioned problems are solved in a pull-based system where a peer wishing to receive a packet from other peer must request it prior to receiving. An obvious weakness with the pull-based system is the issue of dealing with free-riders [63, 86]. A free-riding peer is only requesting and receiving packets from other peers, without uploading anything to others. It is obvious that this affects the performance of the network, because free-riders do not send requested packets to other peers. Another possible weak point in the pull-based system is the packet request overhead. If there are lots of free-riders in the system or overlay network failures are very common, it is possible that multiple peers are requesting the same packet multiple times.

A P2P media streaming solution based on the BitTorrent protocol is proposed in [120]. The rarest-first chunk downloading policy is replaced by a policy where peers first download chunks that will be consumed in the near future. This is done by introducing a sliding window and preventing peers from requesting chunks located outside their current window. The tit-for-tat peer selection policy is also modified to allow free tries to a larger number of peers to let peers participate sooner in the media distribution. Another P2P media streaming system based on a P2P file sharing

implementation has been proposed already in [60]. This receiver-driven P2P media streaming system was built on top of the Gnutella protocol and according to the authors the system was first prototype implementation with multi-sender bandwidth aggregation, adaptive buffer control, peer failure detection and streaming quality maintenance functionalities.

However, the data partitioning based on fixed byte ranges is not suitable for streaming a continuous media, which is of variable bit rate nature. A dedicated application is admittedly a better choice for P2P media streaming purposes. The list of existing P2P media streaming systems which were active in September 2010 is available in Appendix B. Good overview and measurement study for one pull-based system, PPLive [107], is available in [46]. Even though some solutions have proven their functionality with wired connections, those might not be suitable for the mobile networking environment.

An RTSP-based mobile P2P media streaming system where original RTP sessions related to a media delivery are split into a number of so-called partial streams, according to a predefined set of parameters, is proposed in [P6]. Proposed approach allows low-complexity re-assembly of the original media session in real-time at the receiving end. Partial streams also help in utilizing the upload capacity with finer granularity than just per one original stream, which is beneficial in mobile environments where bandwidth can be scarce.

At the time of writing this Thesis, standardization work in the P2P media streaming field has been also started. The Peer-to-Peer Streaming Protocol (PPSP)[5] IETF working group will design a protocol for signalling and control between trackers and peers, and a signalling and control protocol for communication among the peers. The former will handle the initial and periodic exchange of meta information between trackers and peers and the latter will control the advertising and exchange of media data availability between the peers. It will be interesting to see is this working group able to create a specification which will be widely used in the P2P media streaming applications in the near future.

[5] http://datatracker.ietf.org/wg/ppsp/charter/

2.3 Summary

In this chapter basic technologies for a large-scale content delivery over the Internet Protocol are briefly presented. The chapter focuses on two approaches which can be used to enhance traditional PTP-based content delivery. Firstly, multicast and broadcast technologies that are efficient ways to deliver the same content to a large user population when native support for PTM delivery is available in the network are briefly covered. Second part of the chapter contains an introduction to the P2P technologies for PTM services using an overlay network built on top of an existing network infrastructure.

In spite of the obvious benefits, IP multicast has not yet been widely utilized. While the multicast deployments might not be solvable globally, limited-scope multicast distribution networks are already utilized for specific applications, like IPTV, and possible can be also used for other services in the near future. P2P-based content delivery is suffering from the diverse spectrum of non-interoperable systems since most of the P2P systems have been initially designed and implemented by individual researchers without any kind of standardization activities. However, usually implementations inside one P2P network are interoperable, due to the publicly available protocol definitions after the initial implementation has been published, and these additional implementations can be used to extend the possible user population, maybe also using devices which where not supported by the initial implementation, for a specific service.

3. CHALLENGES

Large-scale content delivery is a challenging and complex task. The content delivery system should scale to large heterogeneous user population and should be able to provide good user experience regardless of the available throughput and processing capacity and the experienced loss pattern. The content delivery application should be also easy to use and the service should guarantee content persistence.

More detailed information about several key challenges, which have to be taken into account in order to gain massive public support for the content delivery system, are next presented in Sections 3.1, 3.2 and 3.3. Challenges have been divided into three different categories: challenges for application developers, service providers, and end users even though some of the presented problems could be viewed from several perspectives. In addition, some general security considerations are highlighted in Section 3.4.

3.1 Application Developers

Application developers are facing several challenges when trying to implement a large-scale content delivery system. During previous few years all kind of communication between different devices using different access network technologies has become more and more attractive. Users want to use the same tried and tested applications all the time regardless of their location. From the service provider's point of view reliability and scalability are the most important features for the application.

Application developers must also take usability into account. People do not always want to, or even cannot, install the stand-alone client software or share their bandwidth for security reasons, which prevents the effective spreading of new content delivery techniques. WWW browser plugins [87], which extend the browser to work

with additional content types, might be used to overcome the stand-alone client software installation problem.

It is also very important to have an appropriate designing phase before the actual application development is started. If the coding is started in a hurry, it is possible that some important features, like reliability or scalability, cannot be any more enhanced (or it requires large changes in the code base) when the application has been tested for the first time. This will of course have an influence on the achievable performance of the application later on. On the other hand, if the application has a reasonable modular structure, as a result from the designing phase, it is possible to quickly change some specific part of the code or include some new features, like an additional FEC scheme, to the application when needed.

3.1.1 Access Network Independence

There are a couple of challenges when developing applications for pervasive networks. Most normal consumer connections are suffering connection limitations based on Network Address Translation (NAT) because ISPs do not want that users use still relatively small bandwidth for hosting services or using P2P technologies. To avoid connection limitations, especially in a P2P communication, application developers should consider including some kind of NAT traversal technique, such as Traversal Using Relays around NAT (TURN) [101] or Session Traversal Utilities for NAT (STUN) [114], to the application. Due to the increasing use of devices with multiple access network technologies, applications should also be able to handle vertical handover across various technology domains to support seamless mobility.

With P2P media streaming the delay and downlink throughput of the access network affect the user experience, and should be taken into account when designing an application to be used with different access network technologies. The delay has an influence on the buffering period at the beginning of the stream, and if the used stream bit rate is too high, in comparison with the downlink throughput, it is not possible to use the application at all. It is also important to pay attention to asymmetric broadband connections, since lower uplink throughput make things non-optimal, as peers with asymmetrical access tend to use more download bandwidth compared with the provided upload bandwidth.

The downlink and uplink throughput and the average Round Trip Time (RTT) val-

Table 1. *Network characteristics*

	Connection	Throughput (Downlink/Uplink)	RTT
Mobile	EDGE	154 kbps / 77 kbps	623 ms
	UMTS	351 kbps / 123 kbps	135 ms
	HSDPA	888 kbps / 347 kbps	88 ms
Leased line	ADSL	1 Mbps / 512 kbps	48 ms
	LAN	100 Mbps / 100 Mbps	5 ms

ues for Enhanced Data rates for GSM Evolution (EDGE) [2], Universal Mobile Telecommunications System (UMTS) [1], High Speed Downlink Packet Access (HSDPA) [4], Asymmetric Digital Subscriber Line (ADSL) [58] and Local Area Network (LAN) [50] network connections are shown in Table 1. The downlink throughput values for the mobile network connections are measured by downloading a 10 MB file from ftp://ftp.funet.fi/dev/ and using the Wireshark network protocol analyser [136] to calculate the value. The uplink throughput values for the mobile network connections are measured by uploading a 10 MB file to one of our university servers and using Wireshark to calculate the value. With the leased line network connections theoretical values are shown for downlink and uplink throughputs. The average RTT values for all aforementioned network connections (from 100 measurements) to the Finnish Communication and Internet Exchange association (FICIX)[1] are measured to get the access network delay.

If we compare characteristics between a pure wired network and a mobile network, where the mobile connects to an access point which then connects to the wired network, there is a clear difference. Only HSDPA connection provides reasonable good downlink throughput and delay values for P2P media streaming purposes. However, uplink connection will become the bottleneck in this network, to which High Speed Uplink Packet Access (HSUPA) [3] will provide enhancements in the near future. On the other hand, all aforementioned mobile networks are suitable for P2P file sharing when the real-time requirements are not so critical.

[1] http://www.ficix.fi/

3.1.2 Reliability

In a best-effort network, like in the Internet, the successful delivery of a packet to its receivers is not guaranteed at the network layer [52, 143]. Only acknowledgement-based protocol, such as TCP, where the receiver lets the sender know what has been received correctly can provide reliability in the transport layer in this kind of network. TCP is however not suitable for large-scale content delivery and more scalable protocols, like UDP, are used instead. Unreliable transport protocols are able to scale up to large and massive receiver groups, but the guarantee for fully reliable file delivery or good user experience in P2P media streaming must be provided at the application layer. Short introduction to protocol layering and comparison between five-layer Internet protocol stack and seven-layer International Organization for Standardization (ISO)[2] Open System Interconnection (OSI) reference model is available in [66, pp. 48-56].

There are different methods for providing reliability for multicast-based file delivery applications and good user experience for P2P media streaming applications, since they have different requirements from a user point of view. These methods include for example repeated transmission and FEC for file delivery applications and packet interleaving and packet retransmission for P2P media streaming applications. More information on how to provide reliability in a large-scale content delivery is presented in Chapter 4.

3.1.3 Scalability

At the start of a mass media content release there are generally many customers requesting the same content. The traditional client-server based content delivery will easily overload the delivery network and none of the customers will be happy about the QoS. Hence, a service provider should utilise multicast or broadcast delivery, which serves very large user groups without overloading server and network resources. However, multicast-based unreliable content delivery will create another scalability challenge to the content delivery server. How to keep the load of the content delivery server as low as possible and still simultaneously guarantee the reliability?

[2] http://www.iso.org/

Scalability is also problem in a P2P networks. When a peer joins the P2P overlay network, it should not pose an unmanageable burden to the overlay network and its operation. Rather it should provide more resources into the system, such as upload bandwidth or processing power. Scalability in the overlay adaptation, in order to follow the dynamic network conditions, is often achieved with the decentralized distribution of the key components among participating peers to prevent a single point of failure types problems when unexpected peer departures occur. So, the overlay network should be constructed in a self-organized manner to scale well in the dynamic network conditions.

However, often there is a limit when the system cannot grow any more and its functionality starts to drop. This can happen because of the limited network bandwidth or processing power of some important entity, or there are simply too many control messages generated on the network that cannot be processed in time. Good scalability affects also service availability in a way that any participant in the system should not become a bottleneck so that service cannot be provided any more. More information about how to enhance scalability in a large-scale content delivery is presented in Chapter 5.

3.2 Service Providers

Service providers are very interested in all the features provided by the application. Scalability of the system and methods for providing reliability are however features that depend on the decisions made by the application developers during the implementation phase. So from the scalability and reliability point of view the service provider can only select some software and use it within the limits.

One of the most critical challenges for the service provider is the content itself. How to provide persistence in a long-tailed service when it is not reasonable to use the primary delivery mode any more? How to partition content so that network resources are optimally used? How to guarantee content owner rights when distributing content in a digital form? How to provide content integrity in an untrusted environment? Some solutions for the aforementioned questions are next discussed in the following subsections.

3.2.1 Content Persistence

Content persistence is very important issue in a large-scale static content delivery. If the service provider uses multicast-based content delivery, the full copy of the content is available for the clients as long as the service is available. However, after the popularity of the content has decreased below certain threshold value, it is not reasonable to use multicast delivery any more. For example, the threshold can be calculated from the relative costs of multicast and unicast delivery. If the popularity of the content increases again, the service provider can of course schedule a new multicast transmission period to decrease the total amount of data delivered in the network. On the other hand, there might still be a significant amount of clients interested to retrieve the content after stopping the multicast-based delivery. In this kind of long-tailed service, the service provider could for example switch to P2P delivery if multi-technology delivery is supported in the first place.

Because peers in the P2P overlay network may arrive and depart in a very dynamic fashion the service provider must follow statistics to guarantee that a full copy of the content is available among receivers. For example in the Delco system [P2], the service provider can follow P2P statistics from the BitTorrent tracker and control its own BitTorrent seeders to backup the delivery. However, the statistics cannot be trusted with full certainty because there can be badly behaving clients who indicate that they have the content but do not upload it.

Content persistence in a P2P media streaming services is not a big issue for the service provider. In a live streaming service, the content is available as long as the original data source wants to stream the content to the P2P network. When the original data source leaves the network, other peers in the delivery tree or chain will be able to play out the stream until all buffered data is consumed. In a VoD streaming service, the full copy of the content is always available at least from the original data source. In addition to the reception buffer, data storage via caching should be used in the VoD streaming service. When using caches, the VoD data can be distributed away from the original data source. This helps relieving the network load from the original data source, as new peers joining the network are able to download VoD content from multiple sources instead of relying on the original data source.

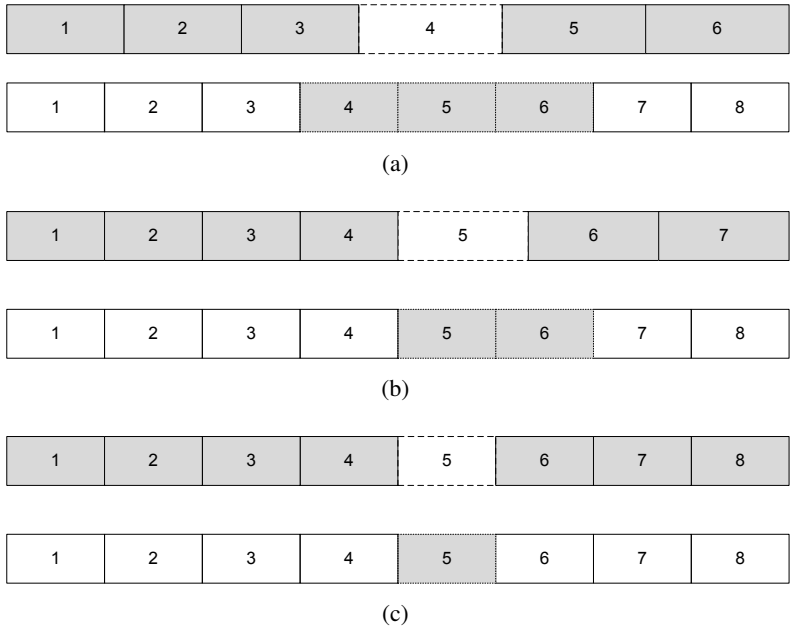

Fig. 9. *Block alignment problem*

3.2.2 Content Partitioning

Content partitioning in a multi-source or multi-technology environment can be a challenging problem for the service provider. In a multi-source P2P content delivery a wide range of block sizes can be used. However, a downloading client becomes a leecher peer when it has at least one complete block, so with a small enough block size the number of alternative source peers will increase faster. In a multi-technology delivery systems, like proposed in [P2], it is crucial that block sizes are adjustable in all delivery technologies. When the block sizes are equal or the bigger block size is a multiple of the smaller block size, all technologies can be optimally used to simultaneously receive data or some technology can be efficiently used to request missing blocks after the delivery is ended with some other technology.

Block alignment problem with a missing block recovery using the Delco system [P2], where FLUTE and BitTorrent are used for large-scale content delivery, as an example is described in Fig. 9. FLUTE's default file partitioning algorithm generates at most two different sizes for the source blocks and the sizes are as close to each other as possible. In Fig. 9a neither of FLUTE's block sizes (the upper chain) are equal to BitTorrent's piece size (the lower chain), and three BitTorrent pieces are needed

for the recovery of one missing FLUTE block and recovery will cause unnecessary duplicate traffic in the network. It should be noted that with FLUTE a data portion is referred to as a block and with BitTorrent as a piece, but conceptually they stand for the same thing, i.e., both represent a chunk of the file. In Fig. 9b smaller FLUTE's block size is equal to BitTorrent's piece size, and two BitTorrent pieces are needed for the recovery. In Fig. 9c FLUTE uses only one block size (enhancement introduced in the Delco system), which is equal to BitTorrent's piece size, and only one BitTorrent piece is needed for the recovery. From this example it is easy to see that appropriate content partitioning is very important in a large-scale content delivery system.

3.2.3 Digital Rights Management

The increased number of households with broadband connectivity and advances in the large-scale content delivery have created both opportunities and threats for the service providers. P2P techniques can be utilised in the delivery systems to increase cost efficiency from the service provider point of view, since end users' devices, together with a small number of service provider's peers, collectively form the delivery infrastructure. However, after the content injection the service provider is not able to restrict the delivery between peers.

Digital Rights Management (DRM) offers protection against the illegal distribution of copyrighted material such as music files or Digital Versatile Disc (DVD) films. For example Windows Media DRM is a proven platform to protect and securely deliver content for playback on computers, portable devices, and network devices [135]. When using DRM in a P2P content delivery [14, 62], the copyrighted material can be delivered with smaller investments into the delivery infrastructure, still preventing the illegal use of the material. DRM can be also used in a multicast or broadcast based content delivery [93, 99] to restrict the content consuming only a licensed user population.

DRM issues have not gained high priority in this Thesis, mainly due to the prototype and proof-of-concept type of implementation work, but for example the Delco system [P2] is designed so that the content distribution and DRM can be separated and various technologies can be used for the content delivery as well as for the DRM.

3.2.4 Content Integrity

Integrity of the delivered content is one of the key issues when delivering content through an untrusted or an unreliable network, like the Internet. If some misbehaving user can modify some parts of the content and share this fake content with the other users, there will be chain reaction of unhappy users in the multi-source environment without proper integrity verification. Typically, the content integrity is verified according to some hash algorithm, for example using MD5 or Secure Hash Algorithm 1 (SHA-1) [90], allowing the receiver to verify the received content by comparing the calculated hash values with the values provided by the sender or the content creator.

Hash values alone are not sufficient enough to guarantee the content integrity if the receiver cannot trust to the hash values provided by the sender or the content creator. This is the case when an attacker can modify both the provided hash values and the actual content. This can be overcome by using digital signatures, i.e., signed hash values for instance by using RSASSA-PKCS1-v1_5 [61], instead of plain hash values. However, techniques relying on public key cryptography require that the public keys of the senders are securely delivered to the receivers. This can be for example achieved with a Public Key Infrastructure (PKI) [49], or by a Pretty Good Privacy (PGP) [23] web of trust, or via pre-distributed public keys.

When delivering content in pieces or blocks it is best to provide hash value for each piece separately. FLUTE provides MD5 hash checksum for the whole file, so when corrupted data is received it is necessary to start the whole downloading process from the beginning as the exact location of the corruption is not known. With the help of piece level verification, only the corrupted piece must be received again which decreases the amount of total data needed for successful delivery. BitTorrent, for example, is effective against Denial of Service (DoS) attack where a hostile peer tries to poison a download with corrupted pieces, since a .torrent file includes a list of piece hashes (SHA-1), which ensures that received and accepted pieces are always correct as long as the .torrent file is not corrupted in the first place.

File download resuming after an accidental application shut-down or system crash can be also done with the help of piece hash values even without any previously saved state information about the progress. For example, in the Delco system [P2] pieces are saved to the file system in their final locations, and it is possible to check, with the help of piece hashes, which pieces are already written and retrieve only the

missing pieces to complete the file downloading.

3.3 End Users

Guarantee for the reliable content delivery and good user experience are the most important issues for the average end user. However, these are provided by the application itself, so major problems and obstacles for the end user deals with the service itself. Service discovery or bootstrapping is the first obstacle when user wants to join a particular multicast or P2P content sharing session. When the user has been able to join the service, the availability of the content and the key components in the content delivery system might influence the Quality of Experience (QoE) of the end user.

3.3.1 Service Discovery

In a multicast-based content sharing system, receivers must join the multicast group in order to receive the data delivered by the server. One way for service discovery in multicast-based content sharing is to define necessary transmission session parameters in a file according to some session description syntax, such as the Session Description Protocol (SDP) [43], which would then be acquired by receivers before the data delivery session begins by means of some transport protocol, like the Session Announcement Protocol (SAP) [44] or HTTP.

A SAP announcement will be sent to a specific well-known multicast group, for example with IP version 4 (IPv4) global scope multicast sessions SAP announcements will be sent to 224.2.127.254, which will be listened by the receivers to get information about available services. With HTTP, the file is of course fetched from some well-known service provider web page describing all available services. Because the original SDP is defined for general real-time multimedia session description purposes, it does not describe all necessary attributes for FLUTE based file delivery sessions. To overcome this problem SDP Descriptors for FLUTE [130] defines required additional SDP attributes for initiating a FLUTE session.

In a P2P content sharing system, P2P client has to know at least one operational node, such as central server, higher level peer or ordinary peer, which to connect to join the existing P2P overlay network. A list of operational peers is often delivered with the

client software and/or retrieved from a web page or another static place. A peer in the list might become unreachable at any moment and it is of course possible that the bootstrapping P2P client does not find any operational peer. This should be rare occasion, but it is possible if the peer list is not retrieved and updated periodically.

3.3.2 Service Availability

Multicast-based content delivery suffers from the limitation that access networks that do not support multicast will not support any multicast-based service at all and it is not possible to receive such service even if desired. To overcome this restriction, the service provider could utilize more than one delivery technology for the same service. If for example P2P delivery is also enabled in addition to the multicast delivery, clients whose access networks do not support multicast can join the P2P delivery. Clients whose access networks support multicast can start seeding to non-multicast clients after they have received some data through the multicast delivery. The service provider must of course have some P2P seeders to backup the delivery, but the server workload will still remain much lower than in the traditional client-server model.

Availability and functionality of the key components, like searching, downloading or indexing, is very important for the P2P networks. Responsibilities of the key components are often distributed among many peers to provide better service availability. Central tracker easily becomes a bottleneck in BitTorrent, since available bandwidth, processing power and memory are often limiting factors. If a tracker goes down, the .torrents depending on the tracker become very quickly non-functional and new peers will not be able to join the distributed download procedure.

Availability of an individual peer is also important, because they can provide more resources on the network. Users are often encouraged for being available on the network, for example by giving better downloading speed when uploading more. NAT traversal also has an effect on peer availability, sometimes peer just cannot be reachable because application does not have sufficient NAT traversal support.

3.4 Security Considerations

Security has not been the central point in this Thesis, so this section introduces some general security threats and counter-measures very briefly. The content delivery sys-

tem can encounter many kinds of attacks, which may target the content delivery infrastructure, the content delivery protocol(s), or the content itself. An attacker may try to compromise the routing infrastructure or the content delivery server by creating congestion so that the user data cannot be delivered in time. It is also possible that someone attacks against some specific functionality in the content delivery infrastructure, like trying to poison the peer list database in the BitTorrent tracker.

Since the session description needed for joining the content delivery session is often delivered out-of-band, it is possible that some other attacker may try to compromise the joining phase by corrupting the SDP description for the FLUTE session or the URL of the tracker in the .torrent file for BitTorrent file sharing session. One way to avoid these kind of problems is to use source authentication when retrieving the session description. This can be done for example by adding authentication data to the SAP announcements or by using HTTP over TLS (HTTPS) [108] when downloading the session description from some WWW server.

Another way to create a DoS attack against legitimate users is to corrupt the meta data needed to actually receive the content. If for example FDT Instance in the FLUTE session or piece hashes in the .torrent file are corrupted the receiver will most likely reject the content even though it has been actually correctly received. In the former case, the meta data integrity can be for example provided at the packet level by using Encapsulating Security Payload (ESP) [64] to check the integrity and authenticate the sender of all the packets being exchanged in the session. It is also possible to use a secure version of the desired protocol if such exists, like HTTPS when retrieving the .torrent file or Secure RTP (SRTP) [16] instead of RTP to provide confidentiality, message authentication, and replay protection to the RTP/RTCP protocols.

Attacks against the content itself can be divided into attacks that try to get access to copyrighted content and attacks that try to corrupt the content. Access control to the content being transmitted can be provided at the file level, for example using DRM (see Subsection 3.2.3) or at the packet level using for instance ESP. It is also very important to validate the content integrity (see Subsection 3.2.4) before consumption to prevent running some malicious code which might be injected within the content. In addition to what is explained in Subsection 3.2.4 it is also possible to provide content integrity at the packet level like the meta data integrity described above.

3.5 Summary

An increasing base of broadband connections, with an increasing average bandwidth, clearly shows that there is great potential for the new types of content delivery services in the near future. For example, in Finland 70% of the households had a broadband connection and 99% of the households had a change to have a leased line broadband connection in April 2009 [67]. Due to the heterogeneous multi-device, multi-access environment of the end users, efficient large-scale content delivery cannot be fulfilled by the intelligent applications only. Hence, advanced solutions require investments also from the service providers and network operators.

Problems and obstacles presented in this chapter give a vision what have to be at least taken into account in a large-scale content delivery. This Thesis will mainly concentrate on the reliability and scalability, but topics which somehow relates to these, like content partitioning, content persistence and service availability, are also further studied. Some of the presented problems, like security and DRM issues, have not been gained high priority in this Thesis and are therefore only briefly covered to get the big picture clearer.

4. RELIABILITY

IP provides a best-effort service for forwarding datagrams from a source to the destination. It does not make any promises about packet delay, packet jitter or packet loss. So, in IP-based applications, failures in the network path between the sender and receiver will cause packet losses. Typically, a burst of packets are lost [13, 20] after which the reception of packets is resumed back to normal, and it is up to the transport protocol or application to deal with the packet losses.

In a P2P communication media is partitioned into pieces or chunks, which allow the media to be received simultaneously from multiple senders. Because these peers may join and leave the P2P overlay network at their own will, media being sent to a receiver may be temporarily interrupted, introducing another reason for packet losses in a P2P system. Hence, in a multi-sender P2P environment two different causes, due to which packet loss may be experienced, can be distinguished: (a) network failure, and (b) peer churn. The amount of actual lost data varies between different P2P protocols and is of course the minimum indexable block even though only one IP packet is lost.

There are different methods to provide reliability for multicast-based file delivery applications and for P2P media streaming applications, since they have different requirements from a user point of view. In a file delivery, one does not care if the data parts arrive in the original order or not, and it usually does not matter how fast the data is delivered. The viewing experience will still be the same once the file is fully downloaded and more focus should be put into optimizing both the total overhead caused by the guarantee of reliability and the load of the server in a file delivery system.

P2P media streaming applications are using client side buffers to eliminate the access network induced delay and jitter. If some packets are lost, those should be recovered before the playback point reaches the gap in the reception buffer to ensure good user

experience. An initial buffering time is very crucial in a P2P media streaming system, because with bigger buffer size it is possible to smooth the variation between packet arrival times and have time for packet loss recovery. On the other hand, the smaller the buffering time is, the faster the playback can be started.

Several methods how to provide reliability for multicast-based file delivery and P2P media streaming applications are next presented in Sections 4.1 and 4.2.

4.1 File Delivery Applications

Bandwidth-efficient delivery of large files to a large number of users has become very important. IP multicast techniques can help to decrease the amount of data transmitted in a system, but reliability becomes a concern, because multicast techniques are commonly based on unreliable transport protocols to scale up to large and massive receiver groups.

Repeated transmissions and FEC are two options to achieve the reliability in the main forwarding path. For the return path there are multiple options such as traditional client-server repair and P2P repair. By finding the optimal combination of repeat transmissions, FEC, and optional file repair data it is possible to decrease the total amount of system-wide data required for a successful large-scale file delivery.

There are many studies concerning reliable multicast-based file delivery on packet erasure channels [78]. For example Almeroth et al. [12] have examined the possibility to use best-effort cyclic multicast to deliver WWW pages. They carried out analysis and simulations to demonstrate the performance of cyclic multicast without FEC. Rodriguez and Biersack [113] extended the cyclic multicast analysis done in [12] and added FEC to the analysis too. Nonnenmacher et al. [91] investigated how FEC can be combined with Automatic Repeat Request (ARQ) to achieve scalable reliable multicast transmission. All aforementioned studies mainly concentrated on the mathematical analysis of reliable multicast and Reed-Solomon was used as a FEC code.

There are also more recent studies considering scalable reliable multicast delivery. For example Neumann et al. [89] introduced large-scale content distribution protocols, which are capable of scaling to massive numbers of users. Their focus was on solutions provided by the IETF RMT working group. Neumann and Roca [88] anal-

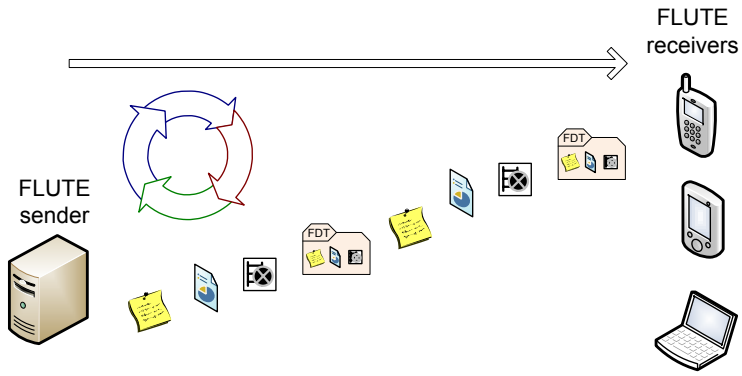

Fig. 10. *Static data carousel with three files described by one FDT Instance*

ysed FEC codes for partially reliable media broadcasting schemes. They studied Low Density Parity Check (LDPC) Staircase and Triangle [112] and Reed-Solomon FEC schemes. Luby et al. [82] [81] introduced and analysed the performance of Raptor FEC codes for reliable download delivery in wireless broadcast systems.

4.1.1 Data Carousel

Repeated transmission of files utilizing a data carousel [78] technique is one way to enable reliable multicast content delivery. The amount of redundant data to achieve a certain level of complete reception can be statistically calculated according to predictable characteristics, such as a packet loss ratio, packet loss distribution, the number of receivers, or the quantity of data. Typical examples of a certain level of complete data would be 99% of receivers achieve complete error-free reception, or 99% of all data is correctly received and decoded by all receivers. Receivers that join a multicast session after its start suffer an initial loss of all data transmitted prior to their joining which complicates this calculation significantly.

Fig. 10 presents a static data carousel with three files described by one FDT instance which is delivered before the files. In this scenario, the content does not change during the FLUTE session and receivers can join and download the content at any time, and all packet losses will finally be recovered - providing that the amount of carousel cycles is sufficiently high. In a dynamic data carousel some of the files in a FLUTE session changes dynamically, and those must be described using a new FDT instance and files must be changed in the carousel. Usually the new file version

supersedes the older version and still receiving clients have to discard the old version and receive the new one instead, because it is not possible to get the old data any more.

Packet loss recovery based on a data carousel is studied in [P1]. In short, it is studied how many carousel cycles are needed to successfully receive the file and what is the effect of changing the encoding symbol length. It is assumed that a single receiver can represent the behaviour of all receivers, which is naturally not the case with the Internet. The assumption is much closer to the truth in environments, where there is only one hop between the sender and the receivers, and the receivers have the similar quality of reception. This is the case for example in DVB-H network when all receivers are located within good network coverage.

4.1.2 FEC Data Carousel

FEC is a mathematically more efficient method of providing redundant data with the potential to achieve better system performance. Adding FEC to the data carousel results in a FEC data carousel [78]. However, depending on the amount of the FEC data and the mean packet loss ratio, several carousel cycles might still be needed.

FEC codes can be divided into systematic and non-systematic codes. With a systematic FEC code, such as Reed-Solomon FEC for small block sizes and Raptor FEC for large block sizes, the first portion of a FEC encoding block consists of source symbols, i.e., the original content items for the given block, while the remaining symbols for the block consist of FEC symbols generated by a FEC encoder. When a non-systematic FEC code is used, all encoding symbols for the block consist of FEC symbols generated by the FEC encoder.

An example of a Reed-Solomon FEC code is presented in Fig. 11. In this example $k = 4$ source symbols together with $n - k = 3$ FEC symbols are transmitted over the packet erasure channel. The receiver can reconstruct the original content when any set of encoding symbols equal in number to the number of source symbols is received.

With Raptor FEC (named MBMS FEC in [6]), the operation of a FEC encoder is divided into several steps. First, the source file is divided into $Z \geq 1$ source blocks and FEC encoding is applied independently to each source block. Next, each source block is divided into $N \geq 1$ contiguous sub-blocks. After that, each sub-block is

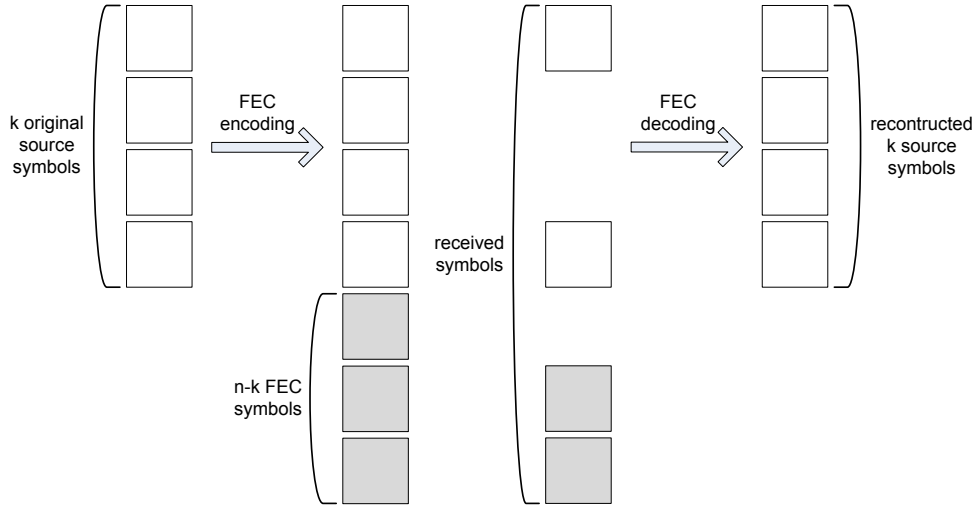

Fig. 11. *Example of a Reed-Solomon FEC code*

divided into K sub-symbols and the mth symbol of a source block consists of the concatenation of mth sub-symbol from each of the sub-blocks. It should be noted that when $N > 1$, then a source symbol is not a contiguous portion of the source file. This happens when the source file size is bigger than the target sub-block size (the recommended value of which is 256 KB). Finally, the FEC encoder generates a desired number of FEC symbols for each source block that consists of K source symbols.

The receiver then receives some set of encoding symbols, slightly more in number than the number of source symbols and feeds those to a FEC decoder. From these encoding symbols, the FEC decoder generates K source symbols, divides each source symbol into N sub-symbols, and composes N sub-blocks which can be concatenated to reconstruct the original source block.

The RaptorQ FEC code described in [77] is a next generation of the Raptor FEC code. The RaptorQ FEC code provides superior flexibility, support for larger source block sizes and better coding efficiency than the Raptor FEC code, which simplify the usage of the RaptorQ FEC code in an object delivery compared with the Raptor FEC code.

Packet loss recovery based on an FEC data carousel is studied in [P1]. Basically the same tests as with the plain data carousel have been performed to study how a different amount of FEC data affect the amount of needed carousel cycles. In

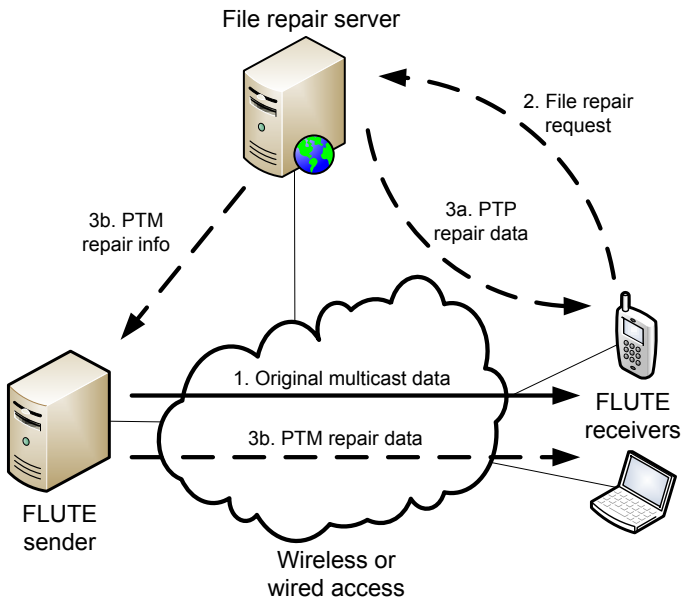

Fig. 12. *PTP and PTM file repair procedure after FLUTE session*

addition, it is also studied what is the effect of changing the source block length, since with bigger source block it is possible to get better protection against losses, but with shorter source block FEC encoding and decoding times are shorter.

4.1.3 Point-to-Point and Point-to-Multipoint File Repair

An option to the carousel type of packet loss recovery is to use some kind of PTP or PTM file repair technique, which can be utilized when encoding symbols are still missing after the FLUTE sender has stopped sending the file. The repairing could happen also during a FLUTE session to decrease downloading time if only a few packets are missing and the next delivery of those packets is suggested being far away in the future.

Both MBMS [6] and IPDC over DVB-H [36] enable PTP and PTM file repair procedures. PTP repair is based on HTTP and PTM repair is based on HTTP and FLUTE. Fig. 12 shows the file repair procedures after the original multicast transmission. First the FLUTE receiver makes a file repair request to the file repair server. The request contains the Uniform Resource Identifier (URI) [18] of the file and the Source Block Number (SBN) and the Encoding Symbol Identifier (ESI) values for the requested

encoding symbols. If the client has several incomplete files or too many encoding symbols are missing from the same file, then it has to make several file repair requests. An example file repair request message is shown in Listing 2.

```
GET /path/repair_script?fileURI=www.example.com/news/
    latest.3gp&SBN=5;ESI=12&SBN=20;ESI=27 HTTP/1.1
Host: mbmsrepair1.example.com
```

Listing 2. *An example file repair request message*

The file repair server can choose between the PTP and PTM repair methods for example based on some algorithm, which uses the cost of transmission of the repair data as efficiency metric. The file repair server behaviour depends on the selected repair method. If the server decides to use PTP repair it responds with *200 OK* message, which includes the encoding symbols the client requested.

If the file repair server decides to use PTM repair, it redirects the clients to a multicast repair session using *302 Found* response message. The temporary URI is given by the Location field in the HTTP response and is the URI of the SDP file of the multicast repair session. In case of PTM repair, the repair information is gathered from the file repair server to the service system. This information consists of the file URIs and/or the Source Block Numbers/Encoding Symbol IDs of the files, which were not delivered completely with the original multicast transmission. According to the information, the service system can schedule FLUTE sender to deliver the repair data through the multicast link. It is implementation specific in which level the repair data is delivered, i.e., the whole file, complete blocks, or only individual encoding symbols.

Simple HTTP-based client-server PTP file repair scheme is mathematically analysed in [P1]. The main focus is to study is it possible to utilise PTP file repair to minimise the total amount of data transmitted in the system when multicast vs. PTP link cost ratio is 1:1.

4.1.4 Peer-to-Peer File Repair

The ultimate option of moving PTP or PTM file repair costs away from service providers and network operators is to base file repair on a P2P scheme. A combination of a P2P overlay network, and a proximity connectivity peer exchange could

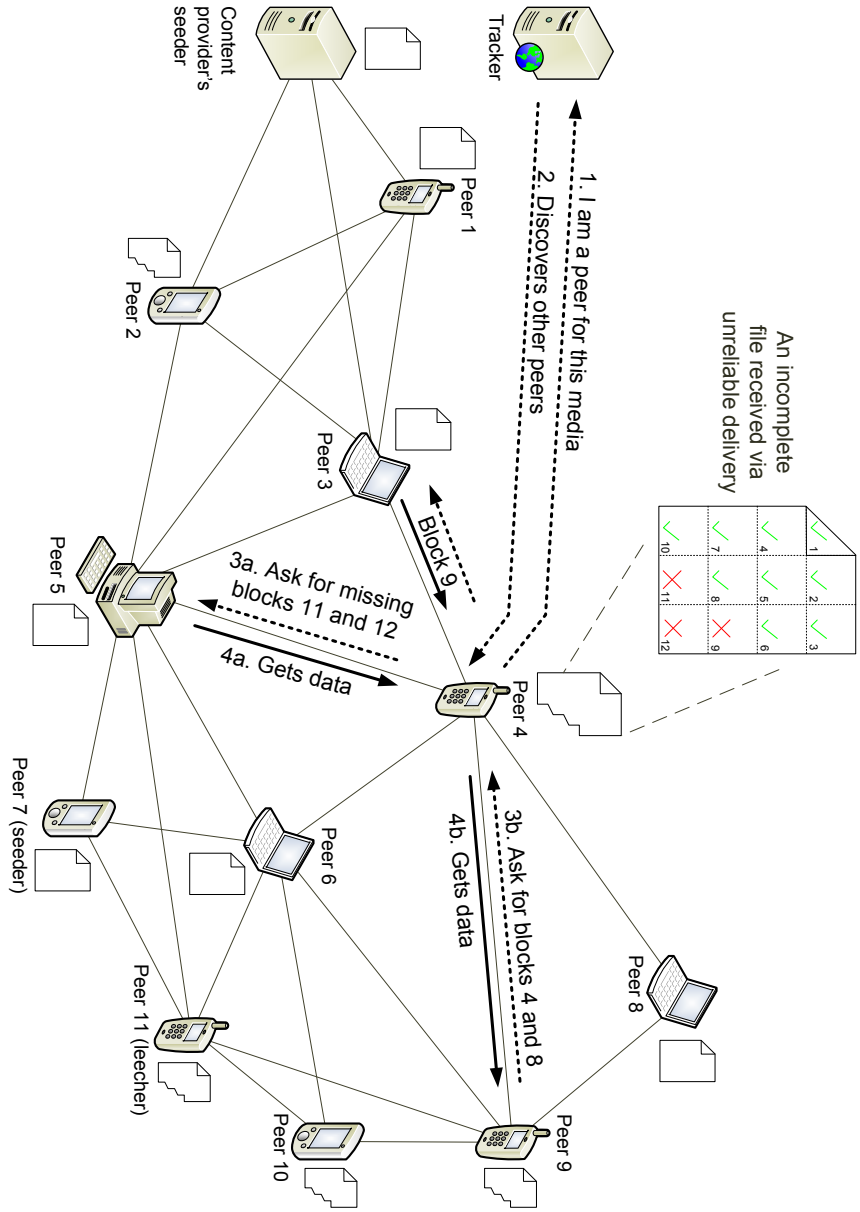

Fig. 13. P2P file repair procedure using BitTorrent

be utilised in the delivery systems to increase cost efficiency. An example P2P file repair scheme is illustrated in Fig. 13, where BitTorrent is used to retrieve missing blocks among other peers after the multicast sender has stopped sending the file. In this example *Peer 4* has an incomplete file and is requesting missing blocks 9, 11, and 12 from *Peer 3* and *Peer 5*. *Peer 4* is simultaneously also providing blocks 4 and 8 for *Peer 9*. The content provider must have some P2P seeders to guarantee the delivery, in a case that there is not a full copy of the file among the clients. Still, the server workload for providing the small subsets of files will remain much lower than in the traditional client-server model.

This kind of file repair scheme is introduced in the Delco system [P2], where multicast and P2P techniques have been combined in a novel way. The system provides the combined benefits of both techniques enabling large-scale content delivery to wired and mobile users with a modest service device infrastructure. In the Delco system P2P file repair can be used to repair/complete the multicast delivery either during the multicast transmission period (simultaneous multicast and P2P file delivery), or afterwards (first multicast file delivery and then P2P file repair). In the first case downloading client should not repair data that is to be received through multicast file delivery.

In addition to the file repair, P2P file delivery can be used to extend the service to clients whose access networks do not support multicast file delivery. Clients whose access networks support multicast file delivery can start seeding to non-multicast clients after they have received at least one block/piece through the multicast file delivery. When using this kind of combined delivery, all receivers can be served, no matter whether they have multicast support or not. However, appropriate content partitioning is very important as explained in Subsection 3.2.2.

4.1.5 Summary

All studied packet loss recovery mechanisms are able to provide reliability in a large-scale file delivery. The results of performance tests carried out for a file delivery system based on the FLUTE protocol are presented in [P1]. In the simulations, the FLUTE sender transmitted data, i.e., an FDT Instance and a single file, in a carousel under uniformly distributed errors and with burst errors. Encoding symbols for each block and also the blocks were sent sequentially, which means that the transmission

of an object was not interleaved. This is not the optimal case with Reed-Solomon FEC, from the packet loss recovery point of view, but it requires less memory from the receiver.

Two types of carousels were studied, a plain data carousel and an FEC data carousel. The plain data carousel used FLUTE with Compact No-Code FEC, and the FEC data carousel used FLUTE with Reed-Solomon FEC, based on Vandermonde matrices [111]. FLUTE performance was measured by the total amount of data the FLUTE sender has to transmit so that a FLUTE receiver receives the file correctly. With a plain data carousel, this is directly related to the number of carousel cycles the sender performs, and with an FEC data carousel also the FEC overhead must be taken into account.

The results from large-scale simulations and related mathematical performance analysis shows that the file delivery performance using a plain data carousel gets quite poor already with low average packet loss. In conclusion, it is obvious that the plain data carousel is not alone sufficient enough for reliable large-scale file delivery, but is of course a better option in comparison with the traditional client-server type of content delivery. The effect of adding even a small amount of FEC data into the data carousel is remarkable both under uniformly distributed errors and with burst errors. When considering the performance of the FEC data carousel without interleaving, it should be noted that the overhead data in one loop increases when the amount of FEC data increases, so the total amount of data averagely needed to transmit describes better the performance of the FEC data carousel.

Since FLUTE is designed for unidirectional transport, the FLUTE sender is not aware of the receiving status of the FLUTE receivers. Thus, it is very important to have a proper configuration at the FLUTE sender to deliver reception guarantees with optimal data expense from the system point of view. Based on the encoding symbol and source block length tests best performance was found when the encoding symbol length was at the maximum length for an encoding symbol carried in the FLUTE packet which kept the complete IP packet length within the link Maximum Transmission Unit (MTU). Likewise, better performance with the FEC code was shown using large source block lengths.

Based on the simplified error reception and distribution model presented in [P1], it is illustrated that the simple HTTP-based client-server PTP file repair is optimal only

for small groups when the total amount of data, which is transmitted in the system and needed for reliable file delivery, is used as a critical factor. However, in some case it might be so that multicast link cost is many times higher than PTP link cost, thus the total amount of data is not the most critical factor any more. The relative costs of PTP and PTM transmission will affect the optimal balance between PTP repair and multicast repeat transmission. However, the detailed analysis of this subject would only be possible using solid cost metrics which are currently unavailable for the DVB-H and MBMS bearers referenced.

The P2P file repair poses new questions for optimizing the multicast-based file delivery. For example, how well must the multicast delivery succeed in order to reach the point where the receivers are able to repair the missing data from each other? To be able to answer the question, a large-scale laboratory and field testing with the Delco system [P2] is needed.

There are a lot of options for the service provider how to provide reliability in a large-scale file delivery. However, a plain data carousel is not sufficient (at least for large user population) and the service provider should utilize some of the three enhanced versions of the data carousel, i.e., data carousel with PTP file repair, FEC data carousel or data carousel with P2P file repair. FEC data carousel will be the best choice for most cases when the total amount of data which is transmitted in the delivery system is used as a critical factor. On the other hand, the service provider could also favour plain data carousel with P2P file repair, since most of the repair data will be exchanged between clients and maybe only one carousel cycle is enough to deliver sufficient amount of the data among the clients.

4.2 Peer-to-Peer Media Streaming Applications

P2P media streaming applications are gaining more and more users around the world. These applications allow end-users to broadcast content throughout IP-based networks in real-time without the need for any special infrastructure, since the user's device, together with all other peers, collectively forms the infrastructure. Furthermore, dedicated servers are no longer required since every peer can serve data to other peers. This is in contrast to a service like YouTube which still requires content to be uploaded to a central server first.

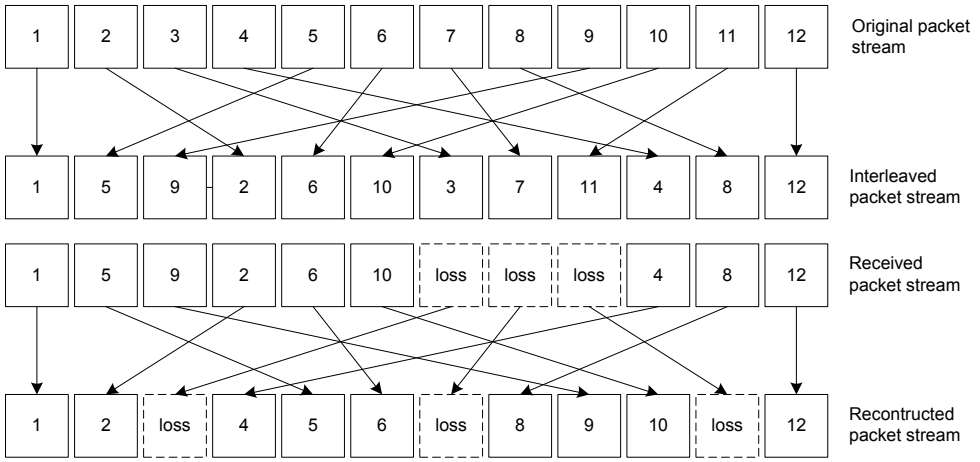

Fig. 14. *Sending interleaved packet stream*

Since most of the existing P2P media streaming applications are using unreliable transport protocols, reliability and good user experience must be guaranteed above the transport layer. Packet interleaving, packet retransmission or FEC can be used either in isolation or together to achieve the wanted reliability level or user experience.

4.2.1 Packet Interleaving

Packet interleaving is one way to mitigate packet loss bursts. Fig. 14 shows an example of interleaved packet stream. Interleaving should be done with a well-known mixing scheme, for example with a fixed block size and the first and last packets of the block in the original position, so that the receiver knows when it has received all packets belonging to a block and can reconstruct the original packet stream. In the example, the sender is sending 12 packets in a mixed order and three packet is lost during the transmission. When the receiver has reconstructed the packet stream a single gap is divided into three smaller gaps and the quality of the media is improved.

An obvious weakness of packet interleaving is that it only smoothes burst errors, it does not provide means to improve reliability. On the other hand, packet interleaving does not increase the used bandwidth so it is a good option to enhance user experience in the application which does not require 100% reliability, like VoIP applications such as Skype [121].

4.2.2 Packet Retransmissions

To ensure seamless media playback in a video streaming application the data should not have interruptions. Therefore, if some data packets are missing, those should be retransmitted before the playback point reaches the gap in the reception buffer. In traditional streaming applications based on RTP/RTCP special mechanisms are available for the retransmission of lost packets. This procedure consists of signalling the losses by the receiver and retransmitting the lost packets by the sender. One method to signal packet losses to the sender is using the generic Negative Acknowledgement (NACK) RTCP Receiver Reports (RRs), as specified in [95]. In addition [109] specifies how the lost packets can be retransmitted by delivering them in a separate stream.

In P2P media streaming the receiving peer does not necessarily know which one of the two options, i.e., network failure or peer churn, is causing the experienced packet loss. Therefore, the traditional RTCP-based packet loss signalling is insufficient in the P2P media streaming case. This is due to the fact that when a sender completely leaves the network, it does not make sense to keep sending generic NACK RTCP RRs to signal lost packets since they will never be received by the departed sender. Another weak point with the RTCP-based packet loss signalling is the fact that a peer does not explicitly request retransmission; it only signals which packets have been lost to the sender, and it is up to the sender to decide how to deal with this information.

In order to allow the scalable and efficient retransmission of lost packets in the presence of network failure and unexpected peer churn, a simple two-stage packet loss recovery mechanism for RTSP-based mobile P2P media streaming system [P6] is proposed in [P7]. Scalability is necessary to prevent the ripple through effect of retransmission requests and redundant retransmissions in case many peers start simultaneously requesting lost packets due to a single cause, like a failing or departed peer.

The proposed packet loss recovery mechanism is illustrated in Fig. 15 where *Peer Y* has experienced packet losses from the *Video 0* sender and requests retransmission of the lost packets from the alternative source peers. After a special time-out value has expired, the signalling of packet losses using RTCP is started (stage one). Firstly, the packet losses are signalled back to the peer it was expected to receive the missing packets from, i.e., *Video 0* sender in Fig. 15, by sending generic NACK RTCP RRs.

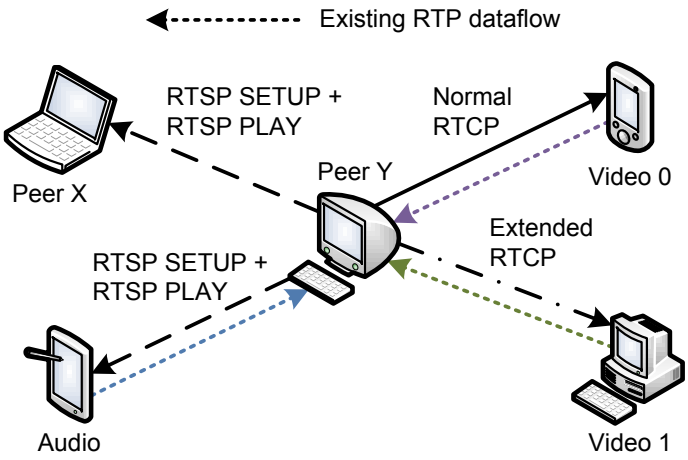

Fig. 15. *Scalable packet loss recovery based on RTCP and RTSP*

This can be considered as a normal operation according to [95]. Alternatively or simultaneously, a similar RTCP RR may be sent to other peers serving different partial RTP streams (see Section 5.3 and [P6] for more information about the partial RTP stream concept) from the same RTP session, i.e., *Video 1* sender in Fig. 15. Note that this extends the scope of the original RTCP specification beyond its normal use.

If the packets that have been signalled as lost using RTCP do not arrive within a certain waiting time, the receiving peer selects a new candidate source peer and starts to request the retransmission of the lost packets by means of RTSP (stage two). The candidate source peers can be peers that are already serving the receiving peer with other media streams of the same service, for example the *Audio* peer in Fig. 15, or completely different peers, like *Peer X* in Fig. 15. Note however that these cannot be peers that are already serving other partials from the same RTP session, since in that case RTCP RRs should be used as described above. Retransmissions from the new peers are explicitly requested by setting up a new RTP session using *RTSP SETUP* and *RTSP PLAY* messages. However, the current syntax of the *Range* header field does not allow requesting the retransmission of individual packets or a limited set of packet ranges; it allows time-based ranges but these do not suffice to uniquely identify single packets. Due to this, a new *Packet-Range* header field is defined in [P7] to make the RTSP-based packet loss signalling possible.

The new RTP session can either be set up permanently, or used only to retrieve miss-

ing packets. In the former case, the old existing RTP session is discarded and playback resumes with the new peer possibly after individual missed packets have been received. In the latter case, the new session will be torn down using the *RTSP TEARDOWN* message after the lost packets have been retransmitted. In order to improve the speed of retransmissions, backup RTSP sessions can be kept open, without actual streaming taking place, just for the purpose of the requesting retransmission of lost packets in case other peers fail. This allows faster error recovery, since the set up time is eliminated from the retransmission procedure. Alternatively, similar performance improvements can also be realized by pipelining the *RTSP SETUP* and *RTSP PLAY* messages as specified in [8] and [118].

In combination with the aforementioned two stages, a special signalling mechanism in order to avoid multiple peers, downstream from the point of failure in a broken path, from signalling lost packets due to the same root cause, such as a malfunctioning peer, or a peer departed in an uncontrolled way or cut off from the network connection, is devised. The proposed mechanism includes two aspects. The first aspect is the signalling of pending retransmission requests downstream, i.e., in the direction of the data flow from the original data source to the leaf nodes. Hence, in the absence of packets, the information is signalled to indicate to the receiving peers that the sender is aware of the losses. Secondly, it includes the signalling of the recovered packets both downstream and upstream. This information can be utilized by the recipients to efficiently reconnect to other peers which have signalled that they have resolved the problem.

4.2.3 Forward Error Correction

The use of FEC in a P2P media streaming poses new requirements for the system design. It is possible to send the FEC data out-of-band using TCP before actual media streaming like in [45]. This kind of solution which separates the FEC packets from the original media packets and creates FEC packets when the media content is created is far from being the most optimal solution. However, the paper claims that traditional packet retransmission is not a suitable packet loss recovery method in P2P media streaming, and therefore FEC is preferred. On the other hand, based on a mathematical analysis a threshold value based on the network load is proposed for switching from the FEC usage to packet retransmissions to avoid congestion because

of FEC overhead.

A better approach is to put FEC layer between transport layer and application layer as specified in [6] for the streaming delivery method or in [71] for multicast-based video delivery over Wireless Local Area Networks (WLANs). The FEC encoding block size is very important when FEC encoding and decoding are done on-the-fly. With a large block size FEC decoding might introduce too much extra delay, in addition to the normal buffering delay, compared with streaming without FEC. However, FEC provides fast packet loss recovery and with appropriate block size, i.e., some fraction of the buffer size, FEC can be efficiently used to enhance user experience in P2P media streaming. On the other hand, unnecessary FEC overhead eats up bandwidth which could be also used to provide more source bytes to improve the stream quality to clients that experience less losses than the lost bandwidth.

Another obstacle in a multi-source P2P environment is how to divide FEC data generation among source peers. In the simplest model, all source peers will generate FEC data based on the whole block and the requesting peer will request the same packet(s) for all blocks from the same peer, for example packet number one for from peer one, packet number two from peer two and so on, like in [39, 137]. The receiving peer can reconstruct the block when it has received k packets from different peers. In this kind solution, the missing packets can be fully recovered when the number of leaving peers plus the number of lost packets is smaller than $n-k$. However, the number of candidate source peers can be limited if a high number of peers suffer from a high packet loss ratio and are not able to recover the block and produce an FEC encoding block for sharing purposes. It might be also possible to store only a subset of the generated FEC packets, in addition to the original media packets which are needed for the local playback, if the amount of peers in a streaming service is reasonably high to ensure the FEC data availability from multiple peers in every situation.

It is obvious that more advanced model for FEC data generation among source peers is needed. One possibility, discussed in our project meetings, could be a distributed Reed-Solomon FEC code where the generator matrix is split among peers and each peer generates FEC symbols based on different part of the source block. However, both mathematical analysis and implementation level verification are needed to show that it is possible to recover the block with one FEC decoding on the receiving side even though different generator matrices are used on the sending side. Some hints towards an advanced model can be also retrieved from [69], where Reed-Solomon

erasure resilient codes system is designed to operate in a P2P networking environment. However, in this system the number of FEC symbols is much larger than the number of source symbols, which might introduce too much overhead from the P2P media streaming point of view.

4.2.4 Summary

As mentioned earlier all presented mechanism can be used to achieve the wanted reliability level or user experience in P2P media streaming applications. Packet interleaving is however not studied in detail in this Thesis. This is due to the fact that packet interleaving is not alone sufficient in a video streaming system, like presented in [P6]. Additionally, the needed RTP operations in [P6] are provided by slightly extended GNU ccRTP library [40] and this kind of feature is not provided in the library.

Laboratory tests with the RTSP-based packet loss recovery mechanism, presented in [P7], have shown that it is beneficial to try to ensure seamless media playback using packet retransmissions. With churning peers, few packets remain lost with the packet loss recovery. This loss is due to the fact that also some of the selected new source peers departs before sending all of the requested missing packets and the small ten seconds initial buffering time does not allow to request some of the missing packets again. With a 1% uniformly distributed packet loss, retransmitted packets are also affected by the packet loss which causes some amount of packets to remain lost also with the packet loss recovery. Full implementation level support for the two-stage packet loss recovery mechanism is still needed to be able to verify the operation of the packet loss recovery mechanism and to be able to fine tune all parameters in the system to maximize the quality and minimize the data delivered in the network.

FEC is especially useful in the unidirectional transmission, when no feedback channel can be used to inform the sender about packet losses, or when there is a low-latency requirement and the initial buffering time is very small in order to allow fast playback at the receiving end. Small initial buffering time gives less time to react to changing network conditions and there might not be enough time for packet retransmissions, requiring other means to provide robustness against packet losses. The use of FEC in a P2P media streaming is only shortly discussed in this Thesis based on a preliminary study. However, FEC could improve the robustness against packet losses

in the proposed RTSP-based mobile P2P media streaming system [P6], [P7] and is therefore an interesting topic for the future developments.

An additional way to provide robustness against packet losses in a P2P media streaming environment is to utilise application layer network coding. Network coding [10] has been originally proposed to enhance multicast delivery, and its suitability for P2P media streaming is studied for example in [131] and [72]. Network coding is not further studied in this Thesis, but it would be interesting to see is it possible and beneficial to integrate application layer network coding to the RTSP-based mobile P2P media streaming system proposed in [P6].

5. SCALABILITY

Traditional content delivery based on the client-server model will easily overload the delivery network and hence more scalable solutions should be utilized in a large-scale content delivery. On the file delivery side it is possible to enhance scalability by using IP multicast, which however requires application layer packet loss recovery. The use of FEC codes is a classical solution to improve the reliability of multicast and broadcast transmissions over a packet erasure channel. However, FEC encoding on-the-fly increases the load of the server and it may decrease the overall performance of the file delivery system.

Scalability is also problem in a P2P networks. A P2P overlay network should be constructed in a self-organized manner, taking location awareness into account, to scale well in the dynamic network conditions. Scalability in the P2P overlay maintenance, in order to follow the dynamic network conditions, is often achieved with the decentralized distribution of the key components among participating peers to prevent a single point of failure types problems when unexpected peer departures occur. Data partitioning is another important scalability issue especially in P2P media streaming applications. When a peer joins the P2P media streaming service, it should be able to provide upload capacity, instead of only downloading the data, regardless of its available upload bandwidth.

Next, a couple of ways to enhance scalability in a large-scale content delivery are presented. First storage format for pre-composed source symbols and pre-calculated FEC symbols to avoid source symbol construction and FEC encoding on-the-fly in a file delivery server is highlighted in Section 5.1. After that some important aspects which can be used to provide more scalable P2P media streaming systems are discussed in Sections 5.2 and 5.3.

5.1 FLUTE Server File Format

The media container file format is an important element in the chain of multimedia content creation, manipulation, transmission and consumption. The file format comprises means of organizing the generated bit stream in such way that it can be accessed for local decoding and playback, transferred as a file, or streamed, all utilizing a variety of storage and transport architectures. The ISO base media file format [55] is a base format for many different media file formats. For example MP4 file format, Advanced Video Coding (AVC) file format [54] and 3GPP file format [7] are based on the ISO base media file format.

The ISO base media file consists of metadata and media data that are enclosed in separate boxes, the movie box (*moov*) and the media data box (*mdat*), respectively. The movie box may contain one or more tracks and each track resides in one *trak* box. A media track refers to samples formatted according to a media compression format and its encapsulation to the ISO base media file format. A hint track refers to hint samples, containing cookbook instructions for constructing packets for transmission over an indicated communication protocol. In addition to timed tracks, the ISO base media file can contain any non-timed binary items. The *meta* box may list and characterize any number of binary items that can be referred to and each one of them can be associated with a file name and unique item identifier.

The FLUTE server file format consists of features that are a part of Edition 3 of the ISO base media file format and Amendment 1 for it [56]. Files intended for the delivery are partitioned into several source blocks, and each source block is then stored as a file reservoir item in a media container file. For each source block additional FEC symbols can be pre-computed and stored as a FEC reservoir item. The actual transmission is controlled by File Delivery (FD) hint tracks containing instructions that ease the encapsulation of source and FEC symbols into FLUTE packets.

Fig. 16 shows an example media container file with one source file. In this example, each source block consists of more than one sub-block, so a source symbol is not a contiguous portion of the source file. Consequently, it is not possible to include source symbols by reference to the original source file. Instead, the media container file contains three file reservoirs labelled File reservoir 1, 2, and N, and an equal number of FEC reservoirs labelled FEC reservoir 1, 2, and M. In a general case, any number N file reservoirs and M FEC reservoirs can be stored in a media container

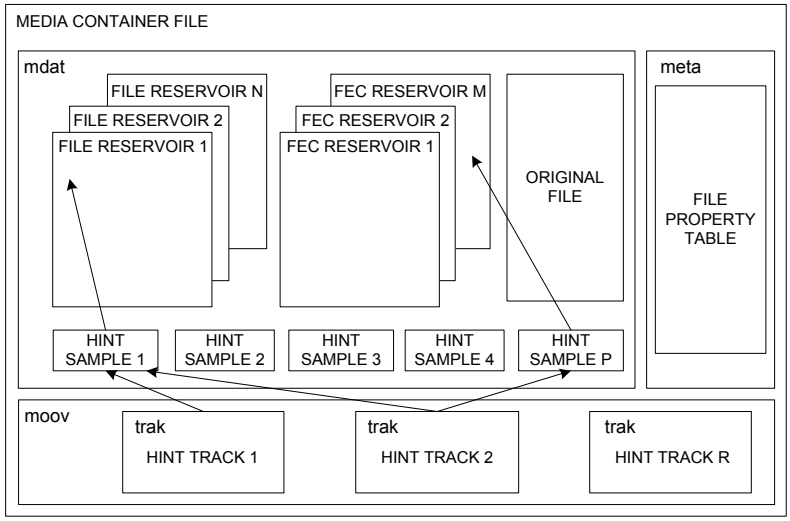

Fig. 16. *An example media container file*

file, and typically $N = M$. When the media container file is formatted according to the ISO base media file format, each file reservoir and FEC reservoir is a binary item of the ISO base media file format.

The file property table can be formatted similarly to the FD Item Information Box of the ISO base media file format. It contains an association meta data to identify items that are file reservoirs and items that are FEC reservoirs. In addition, the association meta data logically links each respective pair of a file reservoir and FEC reservoir with each other, i.e., the source symbols of a source blocks and the FEC symbols derived from the source block. In practice, the association meta data can be a loop or a table of partition entries as described subsequently.

The media container file may additionally comprise any number of hint tracks for instructing in deriving packets from file and FEC reservoirs for file delivery. The hint tracks can be formatted according to the FD hint tracks of the ISO base media file format. File and FEC reservoirs can be used independently of FD hint tracks and vice versa. The reservoirs aid the design of hint tracks and allow alternative hint tracks, for example with different FEC overheads, to re-use the same FEC symbols. They also provide means to access source symbols and additional FEC symbols independently for post-delivery repair, which may be performed over FLUTE or out-of-band via another protocol. In order to reduce complexity when a server follows hint track instructions, hint samples refer directly to the data ranges of the items to be copied

into the hint samples.

The support for file delivery is designed to optimize the server transmission process by enabling FLUTE servers to follow simple instructions. It is enough to follow one pre-defined sequence of instructions per channel in order to transmit one session. The file format allows storage of alternative FLUTE transmission session instructions that may lead to equivalent end results. Such alternatives may be intended for different channel conditions because of higher FEC protection or even by using different FEC schemes.

The prototype implementation for a reliable, server-friendly and bandwidth-efficient file delivery system using the FLUTE server file format is presented in [P5]. The prototype system has been used both to evaluate the FLUTE server file format specification during the standardization process and to study the performance issues relating to the on-the-fly source block partitioning and FEC encoding. The impact of a systematic on-the-fly source block partitioning has been studied using Compact No-Code FEC scheme, Reed-Solomon FEC measurements demonstrate the FEC encoding effect when the source block size is relatively small and Raptor FEC measurements show how large source block sizes affect on-the-fly FEC encoding.

5.2 Clustered Overlay Structure

In P2P content distribution, an overlay network is created at the application layer in order to transfer the actual content among peers in the network. A random mesh-based overlay architecture, like in [142] and [84], provides flexibility for handling peer departures, but good general connectivity between peers is not usually achieved. There have been many studies about how to organize peers in an efficient and scalable way. In [126] receivers are organized into a hierarchy of bounded-size clusters and the multicast tree is built based on that. In [70] peers are organized into a directed acyclic graph to enable peers to obtain locality awareness in a distributed fashion. To improve the file sharing performance of the BitTorrent protocol, Clustered BitTorrent (CBT) concept where peers are grouped into clusters according to their proximity is proposed in [140].

Fig. 17 presents an example architecture of a CBT overlay network. The CBT overlay network is composed of a tracker and three distinct types of peers: super-peer, seeder

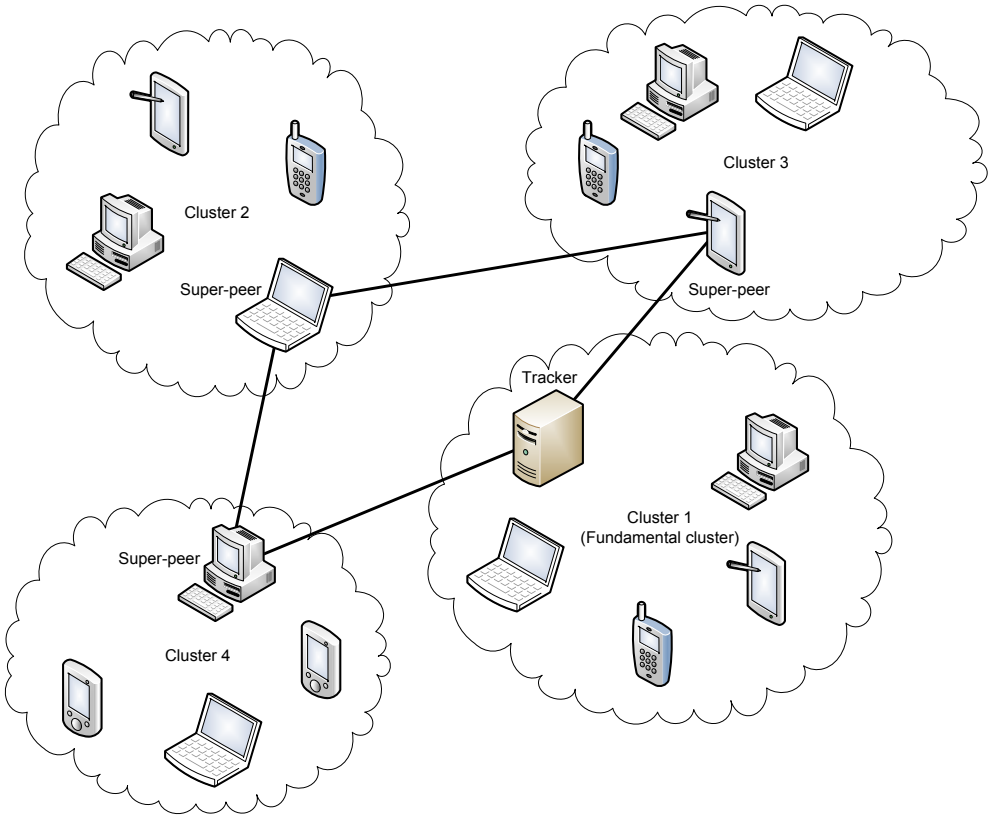

Fig. 17. *Example architecture of a CBT overlay network*

and leecher. The tracker has normal BitTorrent functionality and acts as a super-peer in the fundamental cluster. Periodical monitoring of the super-peer backbone network, to be able to provide precise information for new peers joining the network, is also tracker's responsibility. The super-peer acts as a cluster head, and collects and maintains the state information about all peers in a cluster. Section 5 in [140] presents simulation results to evaluate the proposed system against the original BitTorrent approach. These simulations showed that the CBT system can achieve faster download speed and higher file availability compared with the original system. However, because the lack of CBT implementation large-scale field testing is not possible and simulation results cannot be verified in the real environment.

Similar cluster based overlay network structure for real-time P2P media streaming services is proposed in [P6]. Peers are grouped into clusters according to their proximity in order to efficiently exchange data between peers. For VoD streaming ser-

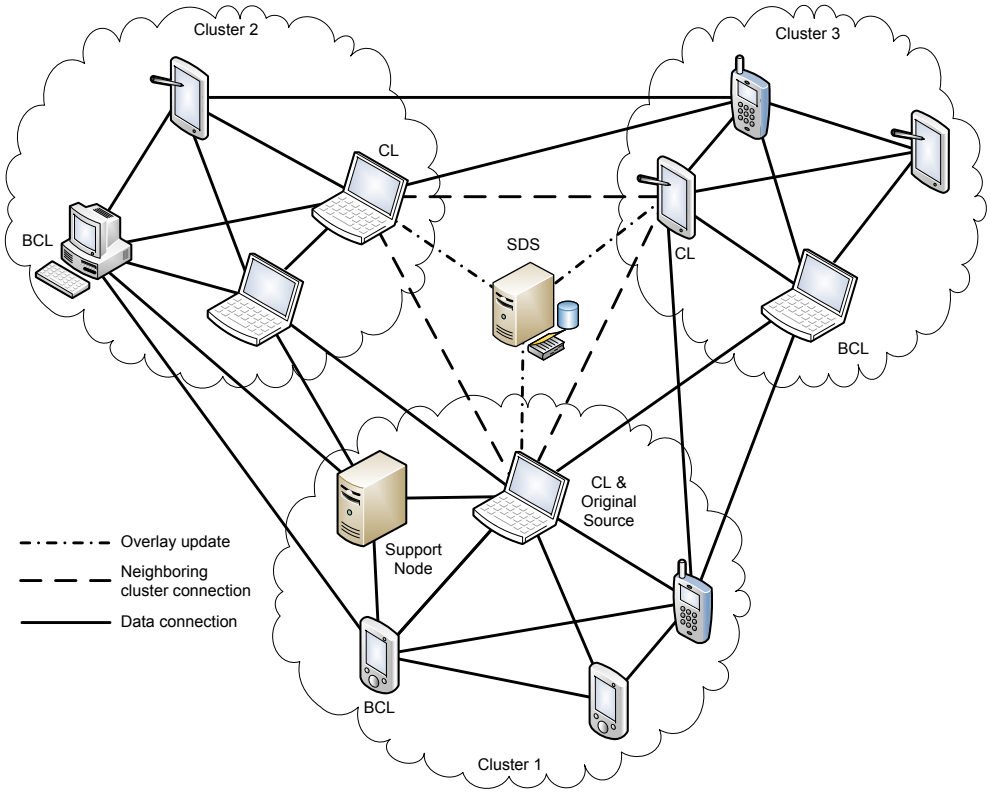

Fig. 18. *Example overlay architecture for P2P media streaming service*

vices, the clusters could be constructed for example based on the interest level for certain pieces of data, so that the peers watching the same part of a video at the same time belong to the same cluster. In live streaming services a cluster can be formed only based on the proximity of peers because all peers are interested in the same data pieces within the same time window. Clusters will also help with scalability issues of peer maintenance. Peers inside a cluster are considered being close to each other and thus communication between peers can be done more efficiently.

The architecture of the overlay network with three clusters sharing a certain streaming service, such as a live stream channel or a VoD movie, is presented in Fig. 18. The cluster concept is implemented with the help of Cluster Leaders (CLs). There is one CL assigned to each cluster with the possibility for one or more Backup Cluster Leaders (BCLs). CLs are used to manage peers inside the cluster and to connect new arriving peers. A new arriving peer can select a suitable cluster according to its best knowledge of locality using RTT values between CLs and itself.

In addition to RTT measurements, location awareness could be also based on for example IP level hop count [28], geographic location [124] or some combination of these three mentioned metrics [140]. IP level hop count is not alone suitable for proximity metric, since with Virtual Private Networks (VPNs) or other tunnelling techniques one hop might actually consist of a large number of hops and the distance could be quite long. Nor does small IP level hop count guarantee small delay, because it does not take connection speed into account. Geographic location is also problematic in the IP level point of view because even if peers are geographically close to each other, the IP level routing path could circulate through a distant router. Hence, only RTT values are used in the system for proximity checks.

It should be noted that for every different streaming service such an overlay network is maintained separately. The Service Discovery Server (SDS) is a central non-mobile server containing information about the cluster hierarchy and the available streaming services in the system. All overlay network operations in the system are implemented using extended RTSP messages. More information about the used RTSP messages and an example message exchange when a peer is participating in a particular service is available in [P6].

One important issue with the clustered overlay architecture is the overlay maintenance. When the CL leaves the streaming service, it needs to be replaced by one of the BCLs. If a cluster does not have an active CL, new peers cannot be accepted into the streaming service. However, this does not affect the data streaming connections between existing peers, because streaming and overlay connections are independent. The merging of two clusters must be done when a cluster becomes too small. If the number of peers is too small, a new joining peer will get a very small list of data sources which makes the functionality less reliable when one of these peers leaves the service. When the cluster grows too large to be handled by a single CL, the cluster should be split into two separate clusters. The existing CL assigns one of its BCLs to become a new CL for the new cluster, and redirects a number of existing peers to the new cluster. This cluster change will not affect the existing data streaming connections and a new joining peer can still prioritize connections within the new cluster. However, new peers should establish data streaming connections also between peers that are located in different clusters to ensure that the new cluster does not become separate island if all ”original” peers leave the streaming service.

In this kind of streaming network, where most of the peers are from the same cluster,

some kind of intelligence to avoid loops is needed. Such loops occur when a sender starts receiving its own data via a number of intermediate peers in the mesh network. Loop detection and/or avoidance mechanism based on the list of ancestors is proposed in [15] and [25]. Similarly, an algorithm based on a streaming path in the form of list of ancestors is used also in [P6]. The path for the data stream in the application level containing peer identifiers which have forwarded the stream is delivered using the Contributing Source (CSRC) list in the RTP packets. This list is then used to avoid accepting connections from peers who are already in the list and for dropping connections if a peer notices that it is in the list.

The main focus in the current system is in the maintenance of the streaming service and the clustered overlay network when malicious interruptions are not considered. However, all P2P systems may suffer from attacks and opportunistic behaviours. Thus, it is quite obvious that a selfish peer can affect and interrupt the expected functioning of the proposed P2P streaming system. Resilience against attacks has not been studied in [P6] or in [P7], so this direction may form the basis for another thesis as a logical continuation for the proof-of-concept implementation work.

5.3 Multiple Stream Approach

Fig. 19 presents a tree layout for a P2P overlay network with the multiple stream approach. A single stream is divided into several descriptions (three descriptions in the figure) and each of the descriptions are then forwarded separately to the network. In this approach a peer receiving all of the descriptions can reconstruct the full quality original stream. If a peer sending out one description leaves the network, the overall stream quality will not collapse remarkably because satisfactory quality can be achieved from the descriptions that still exist. If *Peer 8* for example leaves the network, it affects only *Peer 4* and *Peer 9*, which still are able to play out the stream with the descriptions that still exist during the replacement peer search.

The multiple stream approach helps also to relieve the churning phenomenon. In churning, multiple (probably hundreds) peers rapidly attach and detach the network within a short period of time, making streams rapidly available and unavailable. If we compare the multiple stream approach to the single stream approach in a churning, we avoid the following problematic case. Suppose that a peer sending out full stream leaves the network. If it is the only provider for that stream, the reception of the

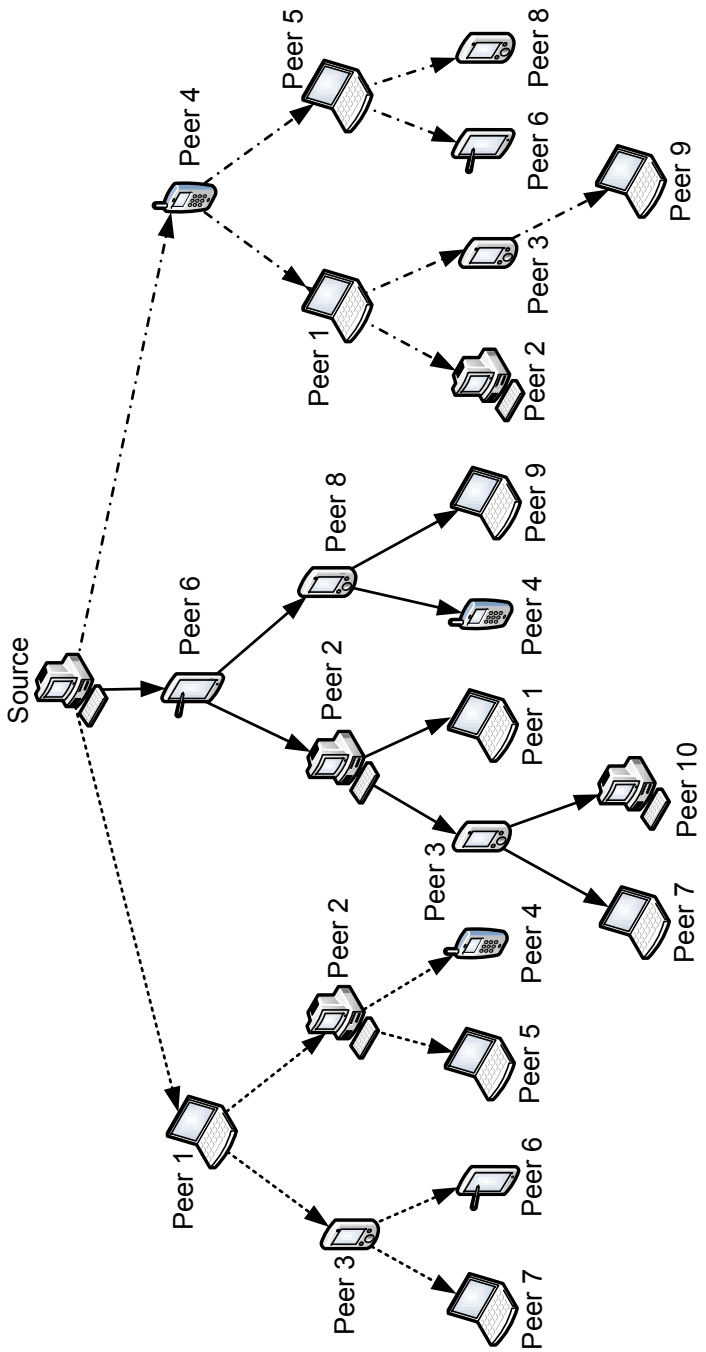

Fig. 19. *Tree-based P2P overlay network with multiple streams*

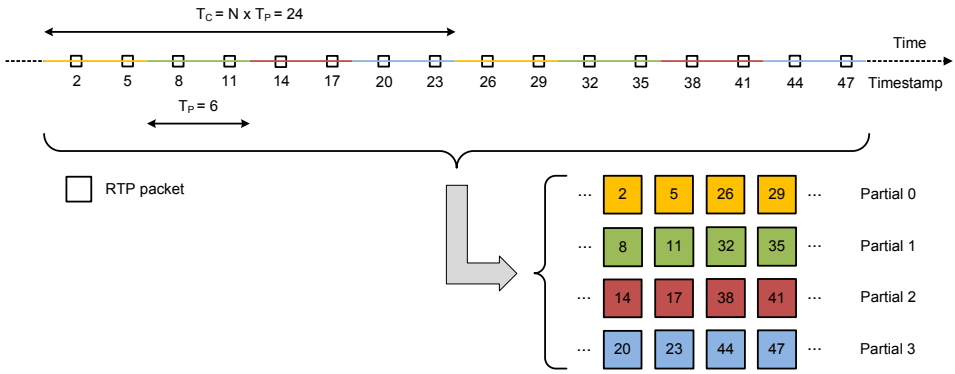

Fig. 20. *RTP stream partitioning*

stream will be cut off from all the peers listening the stream sent by the detached peer. Hence, a large amount of peers joining and departing the network in a single stream system may cause a full loss of stream at times.

For this kind of stream partitioning, there are already some well performing solutions available, such as Multiple Description Coding (MDC) [42] or Scalable Video Coding (SVC) [119] [57], but with the lack of a publicly available implementation. However, the usage of MDC or SVC in the real-time P2P media streaming is an interesting research area as is proposed also in [84] and [86]. Multiple stream approach based on MDC or SVC allows also clients with a low throughput access network connection to play out the stream with the smaller amount of descriptions, like *Peer 8* in Fig. 19, and the overall quality of the stream increases.

Because of the lack of a publicly available MDC or SVC implementation and in order to have unique sending slots for each of the sending peers, a partial RTP stream concept for RTSP-based mobile P2P media streaming system is presented in [P6]. The original RTP sessions related to a media delivery are split into a number of so-called partial streams according to a pre-defined set of parameters in such a way that it allows low-complexity re-assembly of the original media session in real-time at the receiving end.

Assuming a time line of a single RTP session, such as audio, video or subtitle stream of the entire multimedia session, RTP stream partitioning is illustrated in Fig. 20. An RTP session is split into smaller pieces, each consisting of a group of RTP packets, along the time axis. Every piece has a fixed duration T_P which is expressed in time (illustrated by different colours in the figure). The N partial streams are constructed

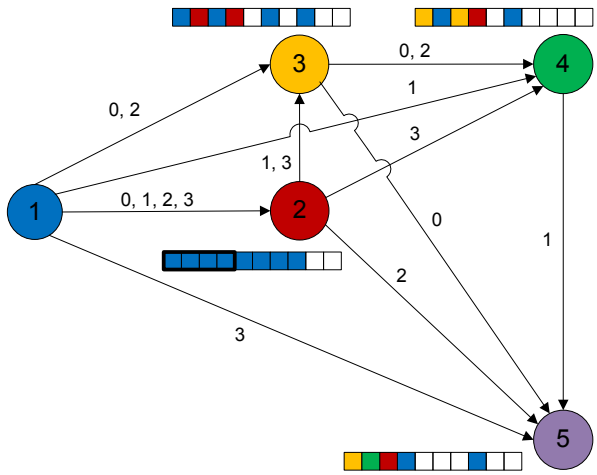

Fig. 21. *Partial RTP stream delivery*

by assigning pieces sequentially one by one into the partial streams *0...N-1* and then continuing again with partial stream *0*, so $T_C = N * T_P$ is used to denote the cycle time. It should be noted that the piece duration T_P should be selected in such a way that it is large enough to contain at least one RTP packet on average. If it is chosen too small, not every piece will have data, which may in the extreme case lead to an empty partial stream. On the other hand, larger cycle times lead to longer start-up times, since at least one complete cycle needs to be buffered before seamless playback can be guaranteed. RTP timestamps and sequence numbers are generated by the original data source and those are delivered unchanged within the streaming service. Due to this RTP packets from multiple partial streams can be reassembled in the correct sequence order at each peer for the local playback and individual packets can be uniquely identified within the streaming service.

A peer may request the delivery of one or more partial streams from another peer. A partial stream is the smallest granularity for media streaming, i.e., a peer may not stream a fraction of a partial stream, and the number of partial streams can be tuned to achieve the target bit rate of a partial stream. Each peer in the network should have enough uplink bandwidth to be able to stream at least a single partial stream; hence the partial stream concept will help in utilizing the upload capacity with finer granularity than just per one original stream. This is beneficial in mobile environments where bandwidth can be scarce.

Fig. 21 illustrates media delivery among four peers. Arrows between each peer de-

note an active RTP session and the direction defines the data flow direction. A sourcing peer can send multiple partial streams to a particular receiving peer. These multiple partial streams could either be streamed in a single RTP session or separate RTP sessions. Peers in the figure are numbered and coloured. The smaller the number is, the earlier the peer has joined the network; in this case peer number one is the original data source. Different colours in the peers buffers show the origin of the received data. For simplicity, the value four is used for the number of partials for each peer. However, the number of partial streams does not necessarily need to be constant throughout the P2P network within a particular streaming service. In the figure peer number two is receiving all partials from the original data source, that is, peer number one, and is forwarding partials to peers number three, four, and five. Peer number four is also receiving partials from peer number three, and peer number five from peers number three and four. More information about the media delivery and data buffering is available in [P6].

Multiple stream approach is also a potential way for implementing an incentive mechanism [96] in a P2P media streaming network. The incentive mechanism can be considered similar to scoring in traditional P2P file sharing networks, where the uploading peers are scored by the amount of data they upload. The better score a peer has, the more privileged are the peer's download opportunities. In a streaming network, a peer uploading more streams has better score, thus being entitled to download more streams and getting a higher quality stream itself. For example, for every uploaded stream, a peer is entitled to download one stream, thus avoiding free-riding and enabling fairness among the peers in the network. However, since system wide download and upload rates match each other, there must be enough peers on the network to provide all download bandwidth by uploading the content.

5.4 Summary

As mentioned earlier in this Thesis multicast or broadcast delivery can be used to serve very large user groups without overloading server and network resources. However, multicast-based unreliable content delivery requires application layer mechanisms to handle packet losses which might compromise the scalability of the file delivery system in some cases. One way to decrease the load of the content delivery and file repair servers is to use FLUTE server file format presented in this Thesis.

The results from the performance measurements using the FLUTE server file format, presented in [P5], showed that the sending time stayed essentially the same regardless of the usage of the media container file with the Compact-No Code FEC and the Reed-Solomon FEC schemes when the maximum source block size is relatively small. However, if a large number of FLUTE receivers are not able to reconstruct the source file completely after the multicast/broadcast file transfer session has been completed, the file repair server might be the bottleneck if media container files are not used. With the FLUTE server file format it is also possible to have a lightweight file repair server, without any FEC capabilities, since the file repair server can use compiling instructions, meta data, and reservoir items to compile the requested data packet set.

The results also showed the importance of avoiding on-the-fly FEC encoding with Raptor FEC. Already with a 100 kB source file, when there is one source block with 1024 source symbols, on-the-fly FEC encoding and sending takes roughly two seconds more compared with the media container file usage. It should be however noted that the used Raptor FEC library might be optimized to perform the FEC encoding faster and to be somehow appropriate for on-the-fly FEC encoding. There exist also other large source block FEC schemes, like LDPC Staircase, which might be appropriate for on-the-fly FEC encoding. To be able to verify this, LDPC Staircase codec available at [94] could be integrated into the current prototype implementation. However, with large source block sizes the file delivery system is much more resistant against burst errors and ought to require less overhead for similar performance.

In P2P content distribution, overlay network structure is very important from the scalability point of view. A random mesh-based overlay architecture provides flexibility for handling peer departures, but good general connectivity between peers is not usually achieved. In this Thesis cluster based overlay network structure for real-time P2P media streaming services is proposed. Peers are grouped into clusters according to their proximity in order to efficiently exchange data between peers and to help with the scalability issues of peer maintenance. Performance evaluation in the laboratory network environment, presented in [P6], has shown that the RTSP signalling overhead needed for the maintenance of the streaming service and the clustered overlay network is quite minimal in comparison with the actual media data.

Another important scalability issue in a P2P media streaming, especially in the mobile networking environment, is the partitioning of the original multimedia data into

smaller parts. The partial RTP stream concept presented in this Thesis allows low-com-plexity re-assembly of the original media session on the receiving side and also help in utilizing the upload capacity with finer granularity than just per one original stream. In addition, the partial RTP stream concept has enabled a fast proof-of-concept implementation due to its low-complexity. However, in order to receive a complete stream, a peer must receive all partial streams in contrast to MDC or SVC when one or more descriptions can be lost. On the other hand, overall system design makes it possible to integrate MDC or SVC implementation into the system as soon as such an implementation becomes publicly available.

First laboratory tests together with the tests in the mobile networking environment, presented in [P6], have shown that the current implementation performs well and offers very low initial buffering times. However, more advanced laboratory tests with different latencies, throughputs and packet losses between peers are still needed to highlight system bottlenecks and usability issues.

6. CONCLUSIONS

The distribution of all kind of digital content is increasingly becoming the primary task of IP-based networks. While the emphasis on P2P file sharing was previously on the exchange of MPEG-1 Audio Layer 3 (MP3) [51] music files with the size of 3-8 MB, the size of files being traded on BitTorrent often exceeds 500 MB [17, 106]. Considering a realistic usage scenario, the delivery of one million copies of one DVD film (size 5 GB) in the P2P network during a period of one month creates an average capacity requirement of 15 Gbps to the distributed delivery system. It is easy to see that a more optimal use of the network capacity is desperately needed; including broadcast and currently unutilized multicast solutions, preferably augmented by P2P-based techniques as is proposed in [P2].

During the previous few years there has been a growing interest in the use of P2P technologies for deploying large-scale live media streaming systems over the Internet. P2P media streaming applications allow end-users to broadcast content throughout the Internet in real-time without the need for any special infrastructure, since the user's device, together with all other peers, collectively forms the infrastructure. In the future, possibly due to the PPSP IETF working group, when there are lots of interoperable P2P media streaming applications which can be used any time regardless of the location, with a huge amount of popular channels, it would be possible that the dominant position of the traditional broadcast television is challenged.

This Thesis studies large-scale content delivery over the Internet Protocol and the target is to create enabling tools for content providers, service providers, network operators and end users to help the distribution and use of all kind of digital content. The research work presented in this Thesis can be divided into two separate areas, (a) file delivery to large user population, and (b) real-time P2P media streaming in a mobile networking environment, focusing on several problems and obstacles which have to be taken into account in order to gain massive public support for the content

delivery system.

Application developers should design the system so that it can scale to large heterogeneous user population and be able to provide good QoS/QoE. Service providers are of course very interested in all the features provided by the application, but one of the most critical challenges from the service provider's point of view is the content itself; since after selecting the used application it can be used within its limits. Major problems and obstacles to the average end user deal with service discovery and service availability. It is also very important to take usability into account when designing the application and the service, since the effective spreading of new content delivery techniques requires easy to access and easy to use services.

6.1 Main Results

The main results of the research work are several prototype or proof-of-concept implementations for both aforementioned research areas. MAD-FLUTE [83] was the first open source FLUTE implementation and it has been utilized around the world for different purposes, for example in IP version 6 (IPv6) SSM testing [129] in a large-scale international IPv6 pilot network [9] and in DVB-H piloting [102]. The Delco content delivery system combined multicast and P2P techniques in a novel way and it is still an exciting and potentially fundamental step in the deployment of large-scale content delivery systems. In addition to these publicly available implementations lot of research and implementation work has been required for the proprietary prototype implementations for file delivery system using FLUTE server file format and for real-time P2P streaming application.

The main research topics have been *reliability* and *scalability*. On the file delivery side it is possible to enhance scalability by using IP multicast, which serves large user population without overloading network resources. However, multicast-based unreliable content delivery requires that packet loss recovery is done at the application layer. As it is shown in this Thesis, there are a lot of options for the service provider how to provide reliability in a large-scale file delivery. FEC data carousel will be the best choice for most cases when the total amount of data which is transmitted in the delivery system is used as a critical factor. On the other hand, the service provider could also favour a plain data carousel with P2P file repair, since most of the repair data will be exchanged between clients and maybe only one carousel cycle is enough

to deliver sufficient amount of the data among the clients. However, application layer packet loss recovery mechanisms might compromise the scalability of the content delivery system in some cases. One way to decrease the load of the content delivery and file repair servers is to use FLUTE server file format for pre-composed source symbols and pre-calculated FEC symbols to avoid source symbol construction and FEC encoding on-the-fly.

In P2P content distribution, overlay network structure is very important from the scalability point of view. In this Thesis cluster based overlay network structure for real-time P2P media streaming services is proposed. Peers are grouped into clusters according to their proximity in order to efficiently exchange data between peers and to help with the scalability issues of peer maintenance. Another important scalability issue in a P2P media streaming, especially in mobile networking environment, is the partitioning of the original multimedia data into smaller parts. This Thesis presents a partial RTP stream concept which allows low-complexity re-assembly of the original media session on the receiving side. Performance evaluation in the laboratory network environment together with the tests in the mobile environment have shown that the current implementation performs well and offers very low initial buffering times compared with most of the other existing P2P media streaming systems. In addition, RTSP signalling overhead needed for the maintenance of the streaming service and the clustered overlay network is quite minimal compared wth the actual media data.

Seamless media playback in the real-time P2P media streaming is one of the most important issues from the user point of view. This Thesis introduces several techniques that can be used to provide good user experience, but the main focus is in packet retransmissions based on RTCP and RTSP protocols. Performance evaluation with the RTSP-based packet loss recovery mechanism has shown that it is beneficial to try to ensure seamless media playback using packet retransmissions. The increase in the RTSP signalling overhead is quite small but the improvement is noticeable. However, full implementation level support for the proposed two-stage packet loss recovery mechanism is still needed to be able to verify the operation of the proposed recovery mechanism.

This Thesis provides separate prototype or proof-of-concept implementations for file-based and streaming-based content delivery. However, it would be worth to study is it feasible to have a single application for file-based and streaming-based content delivery and what kind of trade-offs in repair mechanisms in such an application

could be done.

6.2 Future Development

The author of this Thesis sees three logical directions for the future development. Firstly, in the file delivery side the FLUTE server file format concept introduced in [P5] could be combined also to the Delco system presented in [P2]. FEC is not currently used in the Delco system since clients will use P2P delivery to complete the multicast delivery if needed. However, by finding the optimal combination of repeat transmissions and FEC during the multicast transmission period, the amount of clients with a full copy of the file will be higher, simultaneously decreasing the total amount of system-wide data required for successful file delivery to very large user population. On the other hand, FEC encoding on-the-fly increases the load of the server and it may decrease the overall performance of the file delivery system. By using the FLUTE server file format, this extra server load can be avoided and the overall system performance increased.

Secondly, the real-time P2P media streaming system for the mobile networking environment presented in [P6] and [P7], could be enhanced with several error robustness techniques, such as FEC and multiple stream approach based on MDC or SVC, in addition to the current mechanisms based on packet retransmissions and peer replacement before the reception buffer underflows. FEC provides fast packet loss recovery, but with inappropriate encoding block size it might introduce too much extra delay, in addition to the normal buffering delay. The multiple stream approach based on MDC or SVC provides two important features for the end user. If a peer providing one description leaves the network, it is not so critical to retrieve a replacement peer quickly, since satisfactory quality can be achieved from the descriptions that still exist and it is possible return to the full quality original stream when all of the descriptions are again available. Peers with low throughput access network connections are also able to play out the stream by using only a suitable number of descriptions and the overall quality of the stream increases.

Thirdly, one interesting research area is the integration of P2P media streaming into the social networking to provide one way to share real-time video clips to a limited user group, i.e., to your friends in the social network. This kind of concept for delivering user generated content to a restricted user group is also proposed in [P4]. In

the latter case, the delivery is based on the service provider's broadcast network in comparison with broadcasting the content throughout the IP-based network without the need for any special infrastructure in the former case.

In addition to the aforementioned directions, building blocks provided by this Thesis can be used as a part of other kind of content delivery systems. At the time of writing this Thesis, CDN systems are mainly used to replicate all kind of pre-stored content, like objects embedded in the WWW pages, files, user generated non-real time media streams and VoD content, near to the users to maximize the available bandwidth for accessing the content in a client-server manner. CDN servers are typically located at the edge or access networks so by combining the CDN system to a local FLUTE-based file delivery the data delivered in the core network can be minimized and benefits from multicast-based content delivery can be achieved without support for the multicast delivery in the core network. It might be also feasible to integrate CDN and P2P technologies in a media streaming system, as is studied for example in [73, 138]. However, content replication in the current CDN systems is typically done prior to the consumption, so there might be scalability issues if the replication to hundreds or thousands of CDN servers has to be done in real-time.

Content delivery in this Thesis is based on a server-centric paradigm, where users have references to specific, physical locations where the content (or part of it) can be retrieved from. At the time of writing this Thesis, lots of efforts have been allocated to changing this paradigm into an information-centric or content-centric view. In this new approach, applications will interact with the communication network using a simple request/reply abstraction and the communication network is then responsible for routing the request towards the nearest provider that can offer the content. So, the fundamental principle is that the communication network should allow a user to focus on the content he or she needs without knowing specific locations where the content is actually stored. It will be interesting to see whether this new approach is gaining ground in the near future or is the old server-centric paradigm slightly modified to support changing usage patterns better.

APPENDIX A
ERRATA

We all occasionally make mistakes. The errata below list the mistakes that exist in the published papers.

- [P3] Jani Peltotalo, Jarmo Harju, Alex Jantunen, Marko Saukko, Lassi Väätämöinen, Igor D.D. Curcio, Imed Bouazizi, and Miska M. Hannuksela, "Peer-to-Peer Streaming Technology Survey," in *Proceeding of the Seventh International Conference on Networking (ICN 2008)*, Cancun, Mexico, April 13–18 2008, pp. 342–350. doi:10.1109/ICN.2008.86

  Remarks

  In Section 6, last paragraph, Advanced Video Coding (AVC) should be Scalable Video Coding (SVC), which is the scalable extension of AVC included in the Fifth edition of the AVC specification. Accordingly reference [2] should have been:

  H. Schwarz, D. Marpe, and T. Wiegand, Overview of the Scalable Video Coding Extension of the H.264/AVC Standard, IEEE Transactions on Circuits and Systems for Video Technology, vol. 17, no. 9, pp. 1103–1120, Sep. 2007.

  At the time of writing this Thesis, SVC is also included in the following ISO/IEC specification:

  ISO/IEC, Information technology – Coding of audio-visual objects – Part 10: Advanced Video Coding, International Organization for Standardization / International Electrotechnical Commission (ISO/IEC), ISO/IEC 14496-10:2009, Fifth Edition, May 2009.

- [P5] Jani Peltotalo, Jarmo Harju, and Miska M. Hannuksela, "Reliable, Server-Friendly and Bandwidth-Efficient File Delivery System using FLUTE Server

File Format," in *Proceedings of the IEEE International Symposium on Broadband Multimedia Systems and Broadcasting 2009 (BMSB2009)*, Bilbao, Spain, May 13–15 2009, pp. 1–6. doi:10.1109/ISBMSB.2009.5133753

Remarks

In Section III, Raptor FEC is also a systematic FEC code in a sense that the encoding symbols include the original source symbols composed from the sub-symbols by the FEC encoder. Updated explanation is available in Subsection 4.1.2.

APPENDIX B
EXISTING PEER-TO-PEER MEDIA STREAMING SYSTEMS

Table B.1 summarises existing P2P media streaming systems which were active (at least have a working home page from a Finnish IP address) in September 2010. Systems are also labelled with live and/or VoD tags, but this labelling might be incorrect with some of the listed systems. Only a small subset of the systems have been tested and all systems do not have an English version of the home page, so also additional information available in the Internet has been used in the labelling.

Table B.1. *Existing P2P media streaming systems*

Software	Home Page	Live	VoD
Abacast	http://www.abacast.com/	X	X
ACTLab TV	http://www.actlab.tv/		X
Afreeca	http://afreeca.com/	X	
AllCast	http://www.allcast.com/	X	
BitTorrent DNA	http://www.bittorrent.com/dna/		X
CloneCast	http://clonecast.free.fr/	X	
Coolstreaming Mediacenter	http://www.coolstreaming.us/hp.php?lang=en	X	
End System Multicast	http://esm.cs.cmu.edu/	X	
Freecast	http://www.freecast.org/	X	
Global Media Services	http://globalmediaservices.net/	X	
Itiva	http://www.itiva.com/	X	X
Joost	http://www.joost.com/		X
MaxTV	http://www.max-tv.be/?lng=en	X	
Mediazone	http://www.mediazone.com/index.html	X	X
NiFTyTV Online Television	http://www.niftytv.com/	X	X
Nodezilla	http://www.nodezilla.net/	X	
Octoshape	http://www.octoshape.com/	X	
P2P-Radio	http://p2p-radio.sourceforge.net/	X	
P2PLive	http://www.p2plive.net/	X	
Pcast	http://itv.mop.com/	X	
PeerCast	http://www.peercast.org/	X	
Peerstream	http://www.peerstream.net/	X	
PPLive	http://www.pptv.com/en/	X	X

Continued on the next page

Table B.1. *Existing P2P media streaming systems – continued from the previous page*

Software	Home Page	Live	VoD
PPStream	http://ppstream.com/	X	
QQLive	http://live.qq.com/	X	
RawFlow	http://www.rawflow.com/	X	
ShareCast	http://www.scast.tv/scast/	X	
SopCast	http://www.sopcast.com/	X	X
Stream-2-Stream	http://s2s.sourceforge.net/	X	
StreamAudio	http://www.streamaudio.com/	X	
StreamerOne	http://www.streamerone.com/	X	
Streamer P2P Radio	http://www.streamerp2p.com/	X	
Swarmcast	http://www.swarmcast.com/	X	X
Trevbus	http://www.trevbus.org/	X	
Tribler Streaming	http://tribler.org/trac/	X	
TVU networks	http://www.tvunetworks.com/	X	
Uusee	http://www.uusee.com/	X	
Vatata	http://www.vatata.com/en/	X	X
Veoh	http://www.veoh.com/	X	X
Vuze	http://www.vuze.com/		X
Zattoo	http://zattoo.com/	X	

BIBLIOGRAPHY

[1] 3GPP, "General Universal Mobile Telecommunications System (UMTS) architecture," 3rd Generation Partnership Project (3GPP), TS 23.101, Dec. 2009. [Online]. Available: http://www.3gpp.org/ftp/Specs/html-info/23101.htm

[2] ——, "GSM/EDGE Radio Access Network (GERAN) overall description; Stage 2," 3rd Generation Partnership Project (3GPP), TS 43.051, Dec. 2009. [Online]. Available: http://www.3gpp.org/ftp/Specs/html-info/43051.htm

[3] ——, "Enhanced uplink; Overall description; Stage 2," 3rd Generation Partnership Project (3GPP), TS 25.319, Jun. 2010. [Online]. Available: http://www.3gpp.org/ftp/Specs/html-info/25319.htm

[4] ——, "High Speed Downlink Packet Access (HSDPA); Overall description; Stage 2," 3rd Generation Partnership Project (3GPP), TS 25.308, Jun. 2010. [Online]. Available: http://www.3gpp.org/ftp/Specs/html-info/25308.htm

[5] ——, "Multimedia Broadcast/Multicast Service (MBMS); Architecture and functional description," 3rd Generation Partnership Project (3GPP), TS 23.246, Jun. 2010. [Online]. Available: http://www.3gpp.org/ftp/Specs/html-info/23246.htm

[6] ——, "Multimedia Broadcast/Multicast Service (MBMS); Protocols and codecs," 3rd Generation Partnership Project (3GPP), TS 26.346, Jun. 2010. [Online]. Available: http://www.3gpp.org/ftp/Specs/html-info/26346.htm

[7] ——, "Transparent end-to-end packet switched streaming service (PSS); 3GPP file format (3GP)," 3rd Generation Partnership Project (3GPP), TS 26.244, Jun. 2010. [Online]. Available: http://www.3gpp.org/ftp/Specs/html-info/26244.htm

[8] ——, "Transparent end-to-end Packet-switched Streaming Service (PSS); Protocols and codecs," 3rd Generation Partnership Project (3GPP), TS 26.234, Jun. 2010. [Online]. Available: http://www.3gpp.org/ftp/Specs/html-info/26234.htm

[9] 6NET Home Page. (2010, Sep.). [Online]. Available: http://www.6net.org/

[10] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, Jul. 2000.

[11] Akamai Home Page. (2010, Sep.). [Online]. Available: http://www.akamai.com/

[12] K. C. Almeroth, M. H. Ammar, and Z. Fei, "Scalable Delivery of Web Pages Using Cyclic Best-Effort Multicast," in *Proceedings of the IEEE INFOCOM 1998*, Mar. 1998, pp. 1214–1221.

[13] J. Andren, M. Hilding, and D. Veitch, "Understanding End-to-End Internet Traffic Dynamics," in *Proceeding of the IEEE Global Communications Conference (IEEE GLOBECOM 1998)*, Nov. 1998, pp. 1118–1122.

[14] G. Arora, M. Hanneghan, and M. Merabti, "P2P Commercial Digital Content Exchange," *Electronic Commerce Research and Applications*, vol. 4, no. 3, pp. 250–263, 2005.

[15] P. Baccichet, J. Noh, E. Setton, and B. Girod, "Content-Aware P2P Video Streaming with Low Latency," in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME 2007)*, July 2007, pp. 400–403.

[16] M. Baugher, D. McGrew, M. Naslund, E. Carrara, and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)," Internet Engineering Task Force, RFC 3711, Mar. 2004. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3711.txt

[17] A. Bellissimo, B. N. Levine, and P. Shenoy, "Exploring the Use of BitTorrent as the Basis for a Large Trace Repository," University of Massachusetts Amherst, Technical Report 04-41, Jun. 2004.

[18] T. Berners-Lee, R. Fielding, and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax," Internet Engineering Task Force, RFC 3986, Jan. 2005. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3986.txt

[19] T. Berners-Lee, L. Masinter, and M. McCahill, "Uniform Resource Locators (URL)," Internet Engineering Task Force, RFC 1738, Dec. 1994. [Online]. Available: http://www.rfc-editor.org/rfc/rfc1738.txt

[20] M. S. Borella, D. Swider, S. Uludag, and G. B. Brewster, "Internet Packet Loss: Measurement and Implications for End-to-End QoS," in *Proceeding of the 1998 International Conference on Parallel Processing (ICPP '98)*, Aug. 1998, pp. 3–12.

[21] R. Buyya, M. Pathan, and A. Vakali, *Content Delivery Networks (Lecture Notes in Electrical Engineering)*. Springer-Verlag Berlin Heidelberg, 2008, vol. 9.

[22] B. Cain, S. Deering, I. Kouvelas, B. Fenner, and A. Thyagarajan, "Internet Group Management Protocol, Version 3," Internet Engineering Task Force, RFC 3376, Oct. 2002. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3376.txt

[23] J. Callas, L. Donnerhacke, H. Finney, D. Shaw, and R. Thayer, "OpenPGP Message Format," Internet Engineering Task Force, RFC 4880, Nov. 2007. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4880.txt

[24] G. Camarillo and IAB, "Peer-to-Peer (P2P) Architecture: Definition, Taxonomies, Examples, and Applicability," Internet Engineering Task Force, RFC 5694, Nov. 2009. [Online]. Available: http://www.rfc-editor.org/rfc/rfc5694.txt

[25] Y.-H. Chu, A. Ganjam, T. S. E. Ng, S. G. Rao, K. Sripanidkulchai, J. Zhan, and H. Zhang, "Early Experience with an Internet Broadcast System based on Overlay Multicast," in *Proceedings of the USENIX 2004 Annual Technical Conference*, 2004, pp. 155–170.

[26] B. Cohen, "Incentives Build Robustness in BitTorrent," in *Workshop on Economics of Peer-to-Peer Systems (P2PECON2003)*, Jun. 2003, pp. 116–121.

[27] ——. (2010, Sep.) The BitTorrent Protocol Specification. [Online]. Available: http://www.bittorrent.org/beps/bep_0003.html

[28] L. Dai, Y. Cao, Y. Cui, and Y. Xue, "On Scalability of Proximity-Aware Peer-to-Peer Streaming," *Comput. Commun.*, vol. 32, no. 1, pp. 144–153, 2009.

[29] S. Deering, "Host extensions for IP multicasting," Internet Engineering Task Force, RFC 1112, Aug. 1989. [Online]. Available: http://www.rfc-editor.org/rfc/rfc1112.txt

[30] Delco Project Home Page. (2010, Sep.). [Online]. Available: http://delco.cs.tut.fi/

[31] S. Dixit and T. Wu, *Content Networking in the Mobile Internet*. Wiley-Interscience, 2004.

[32] B. M. Edwards, L. A. Giuliano, and B. R. Wright, *Interdomain Multicast Routing: Practical Juniper Networks and Cisco Systems Solutions*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002.

[33] Elisa Viihde (in Finnish). (2010, Sep.). [Online]. Available: http://www.elisa.fi/viihde/

[34] ETSI, "Digital Video Broadcasting (DVB); Transmission System for Handheld Terminals (DVB-H)," European Telecommunications Standards Institute (ETSI), ETSI EN 302 304, V1.1.1, Nov. 2004.

[35] ——, "Digital Video Broadcasting (DVB); IP Datacast over DVB-H: Set of Specifications for Phase 1," European Telecommunications Standards Institute (ETSI), ETSI TS 102 468, V1.1.1, Nov. 2007.

[36] ——, "Digital Video Broadcasting (DVB); IP Datacast over DVB-H: Content Delivery Protocols," European Telecommunications Standards Institute (ETSI), ETSI TS 102 472, V1.3.1, Feb. 2009.

[37] G. Faria, J. A. Henriksson, E. Stare, and P. Talmola, "DVB-H: Digital Broadcast Services to Handheld Devices," *Proceedings of the IEEE*, vol. 94, no. 1, pp. 194–209, Jan. 2006.

[38] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1," Internet Engineering Task Force, RFC 2616, Jun. 1999. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2616.txt

[39] V. Gau, P.-J. Wu, C.-N. Lee, and J.-N. Hwang, "A Scheme for Peer-to-Peer Live Streaming with Multi-Source Multicast and Forward Error Correction," in *Proceedings of the 2008 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '08)*, Mar. 2008, pp. 2173–2176.

[40] GNU ccRTP - GNU Telephony. (2010, Sep.). [Online]. Available: http://www.gnu.org/software/ccrtp/

[41] B. Goode, "Voice over Internet Protocol (VoIP)," *Proceedings of the IEEE*, vol. 90, no. 9, pp. 1495–1517, Sep 2002.

[42] V. K. Goyal, "Multiple Description Coding: Compression Meets the Network," *IEEE Signal Processing Magazine*, vol. 18, no. 5, pp. 74–94, Sep. 2001.

[43] M. Handley and V. Jacobson, "SDP: Session Description Protocol," Internet Engineering Task Force, RFC 2327, Apr. 1998. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2327.txt

[44] M. Handley, C. Perkins, and E. Whelan, "Session Announcement Protocol," Internet Engineering Task Force, RFC 2974, Oct. 2000. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2974.txt

[45] M. Hayasaka and T. Miki, "Peer-to-Peer Multimedia Streaming with Guaranteed QoS for Future Real-time Applications," *Transactions of Information Processing Society of Japan*, vol. 49, no. 3, pp. 1364–1373, Mar. 2008.

[46] X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross, "A Measurement Study of a Large-Scale P2P IPTV System," *IEEE Transactions on Multimedia*, vol. 9, no. 8, pp. 1672–1687, Dec. 2007.

[47] H. Holbrook and B. Cain, "Source-Specific Multicast for IP," Internet Engineering Task Force, RFC 4607, Aug. 2006. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4607.txt

[48] M. Hosseini, D. T. Ahmed, S. Shirmohammadi, and N. D. Georganas, "A Survey of Application-Layer Multicast Protocols," *Communications Surveys & Tutorials, IEEE*, vol. 9, no. 3, pp. 58–74, 2007.

[49] R. Housley, W. Polk, W. Ford, and D. Solo, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," Internet Engineering Task Force, RFC 3280, Apr. 2002. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3280.txt

[50] IEEE Standards Association, "IEEE Standard for Information Technology – Telecommunications and Information Exchange Between Systems – Local and Metropolitan Area Networks – Specific Requirements – Part 3: Carrier Sense Multiple Access with Collision Detection (CMSA/CD) Access Method and Physical Layer Specifications," IEEE 802.3 Ethernet Working Group, IEEE 802.3-2008, Part 3, Section Two, Jun. 2010.

[51] ISO/IEC, "Information technology – Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s – Part 3: Audio," International Organization for Standardization/International Electrotechnical Commission (ISO/IEC), ISO/IEC 11172-3:1993, Aug. 1993.

[52] ——, "Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model," International Organization for Standardization/International Electrotechnical Commission (ISO/IEC), ISO/IEC 7498-1:1994, Second Edition, Nov. 1994.

[53] ——, "Information technology – Coding of audio-visual objects – Part 14: MP4 file format," International Organization for Standardization/International Electrotechnical Commission (ISO/IEC), ISO/IEC 14496-14:2003, First Edition, Nov. 2003.

[54] ——, "Information technology – Coding of audio-visual objects – Part 15: Advanced Video Coding (AVC) file format," International Organization for Standardization/International Electrotechnical Commission (ISO/IEC), ISO/IEC 14496-15:2004, First Edition, Apr. 2004.

[55] ——, "Information technology – Coding of audio-visual objects – Part 12: ISO base media file format," International Organization for Standardiza-

tion/International Electrotechnical Commission (ISO/IEC), ISO/IEC 14496-12:2008, Third Edition, Oct. 2008.

[56] ——, "Information technology – Coding of audio-visual objects – Part 12: ISO base media file format, AMENDMENT 1: General improvements including hint tracks, metadata support, and sample groups," International Organization for Standardization/International Electrotechnical Commission (ISO/IEC), ISO/IEC 14496-12:2008/Amd.1, Final Proposed Draft Amendment, MPEG document N10249, Oct. 2008.

[57] ——, "Information technology – Coding of audio-visual objects – Part 10: Advanced Video Coding," International Organization for Standardization/International Electrotechnical Commission (ISO/IEC), ISO/IEC 14496-10:2009, Fifth Edition, May 2009.

[58] ITU-T, "Asymmetric Digital Subscriber Line (ADSL) Transceivers," International Telecommunication Union - Telecommunication Standardization Sector (ITU-T), ITU-T Recommendation G.992.1, Jul. 1999.

[59] M. Izal, G. Urvoy-Keller, E. W. Biersack, P. A. Felber, Al, and L. Garcés-Erice, "Dissecting BitTorrent: Five Months in a Torrent's Lifetime," in *Proceedings of the 5th International Workshop on Passive and Active Measurement (PAM 2004)*, Apr. 2004, pp. 1–11.

[60] X. Jiang, Y. Dong, D. Xu, and B. Bhargava, "GnuStream: a P2P Media Streaming System Prototype," in *Proceedings of the International Conference on Multimedia and Expo (ICME 2003)*, Jul. 2003, pp. 325–328.

[61] J. Jonsson and B. Kaliski, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1," Internet Engineering Task Force, RFC 3447, Feb. 2003. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3447.txt

[62] T. Kalker, D. H. J. Epema, P. H. Hartel, R. L. Lagendijk, and M. Van Steen, "Music2Share - Copyright-Compliant Music Sharing in P2P Systems," *Proceedings of the IEEE*, vol. 92, no. 6, pp. 961–970, June 2004.

[63] M. Karakaya, I. Korpeoglu, and O. Ulusoy, "Free Riding in Peer-to-Peer Networks," *Internet Computing, IEEE*, vol. 13, no. 2, pp. 92–98, March-April 2009.

[64] S. Kent, "IP Encapsulating Security Payload (ESP)," Internet Engineering Task Force, RFC 4303, Dec. 2005. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4303.txt

[65] E. Kohler, M. Handley, and S. Floyd, "Datagram Congestion Control Protocol (DCCP)," Internet Engineering Task Force, RFC 4340, Mar. 2006. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4340.txt

[66] J. F. Kurose and K. W. Ross, *Computer Networking: A Top-Down Approach*, 5th ed. Addison-Wesley Publishing Company, 2009.

[67] Laajakaistainfo.fi (in Finnish). (2010, Sep.). [Online]. Available: http://www.laajakaistainfo.fi/

[68] J. Lacan, V. Roca, J. Peltotalo, and S. Peltotalo, "Reed-Solomon Forward Error Correction (FEC) Schemes," Internet Engineering Task Force, RFC 5510, Apr. 2009. [Online]. Available: http://www.rfc-editor.org/rfc/rfc5510.txt

[69] J. Li, "Efficient Implementation of Reed-Solomon Erasure Resilient Codes in High-Rate Applications," U.S. Patent 7,418,649, Aug. 26, 2008.

[70] J. Liang and K. Nahrstedt, "DagStream: Locality Aware and Failure Resilient Peer-to-Peer Streaming," in *Proceedings of the 13th Annual Multimedia Computing and Networking Conference (MMCN'06)*, Jan. 2006, pp. 224–238.

[71] H. Liu, M. Wu, D. Li, S. Mathur, K. Ramaswamy, L. Han, and D. Raychaudhuri, "A Staggered FEC System for Seamless Handoff in Wireless LANs: Implementation Experience and Experimental Study," in *Proceedings of the Ninth IEEE International Symposium on Multimedia*, Dec. 2007, pp. 283–290.

[72] H. Liu, X. Tu, and J. Xie, "Network Coding for P2P Live Media Streaming," in *IFIP International Conference on Network and Parallel Computing 2008 (IFIP NPC2008)*, Oct. 2008, pp. 392–398.

[73] X. Liu, H. Yin, and C. Lin, "A Novel and High-Quality Measurement Study of Commercial CDN-P2P Live Streaming," in *Proceeding of the 2009 International Conference on Communications and Mobile Computing (CMC 2009)*, vol. 3, Jan. 2009, pp. 325–329.

[74] Y. Liu, Y. Guo, and C. Liang, "A Survey on Peer-to-Peer Video Streaming Systems," *Peer-to-Peer Networking and Applications*, vol. 1, no. 1, pp. 18–28, March 2008.

[75] M. Luby and V. Goyal, "Wave and Equation Based Rate Control (WEBRC) Building Block," Internet Engineering Task Force, RFC 3738, Apr. 2004. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3738.txt

[76] M. Luby, A. Shokrollahi, M. Watson, and T. Stockhammer, "Raptor Forward Error Correction Scheme for Object Delivery," Internet Engineering Task Force, RFC 5053, Oct. 2007. [Online]. Available: http://www.rfc-editor.org/rfc/rfc5053.txt

[77] M. Luby, A. Shokrollahi, M. Watson, T. Stockhammer, and L. Minder, "RaptorQ Forward Error Correction Scheme for Object Delivery," Internet Engineering Task Force, Internet-Draft draft-ietf-rmt-bb-fec-raptorq-03, Jun. 2010, work in progress. [Online]. Available: http://www.ietf.org/internet-drafts/draft-ietf-rmt-bb-fec-raptorq-03.txt

[78] M. Luby, L. Vicisano, J. Gemmell, L. Rizzo, M. Handley, and J. Crowcroft, "The Use of Forward Error Correction (FEC) in Reliable Multicast," Internet Engineering Task Force, RFC 3453, Dec. 2002. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3453.txt

[79] M. Luby, M. Watson, and L. Vicisano, "Layered Coding Transport (LCT) Building Block," Internet Engineering Task Force, RFC 5651, Oct. 2009. [Online]. Available: http://www.rfc-editor.org/rfc/rfc5651.txt

[80] ——, "Asynchronous Layered Coding (ALC) Protocol Instantiation," Internet Engineering Task Force, RFC 5775, Apr. 2010. [Online]. Available: http://www.rfc-editor.org/rfc/rfc5775.txt

[81] M. Luby, T. Gasiba, T. Stockhammer, and M. Watson, "Reliable Multimedia

Download Delivery in Cellular Broadcast Networks," *IEEE Transactions on Broadcasting*, vol. 53, no. 1, pp. 235–246, Mar. 2007.

[82] M. Luby and M. Watson and T. Gasiba and T. Stockhammer and W. Xu, "Raptor Codes for Reliable Download Delivery in Wireless Broadcast Systems," in *Proceedings of the Consumer and Communications Networking Conference (CCNC 2006)*, Jan. 2006, pp. 192–197.

[83] MAD Project Home Page. (2010, Sep.). [Online]. Available: http://mad.cs.tut.fi/

[84] N. Magharei and R. Rejaie, "PRIME: Peer-to-Peer Receiver-drIven MEsh-Based Streaming," in *Proceedings of the 26th IEEE International Conference on Computer Communications (INFOCOM 2007)*, May 2007, pp. 1415–1423.

[85] Maxinetti PCTV (in Finnish). (2010, Sep.). [Online]. Available: http://pctv.maxinetti.fi/

[86] J. D. Mol, D. H. P. Epema, and H. J. Sips, "The Orchard Algorithm: Building Multicast Trees for P2P Video Multicasting Without Free-Riding," *IEEE Transactions on Multimedia*, vol. 9, no. 8, pp. 1593–1604, Dec. 2007.

[87] Mozilla Firefox Plugins. (2010, Sep.). [Online]. Available: https://addons.mozilla.org/en-US/firefox/browse/type:7

[88] C. Neumann and V. Roca, "Analysis of FEC Codes for Partially Reliable Media Broadcasting Schemes," in *Proceedings of the 2nd International Workshop on Multimedia Interactive Protocols and Systems (MIPS'04)*, Nov. 2004, pp. 108–119.

[89] C. Neumann, V. Roca, and R. Walsh, "Large Scale Content Distribution Protocols," *ACM SIGCOMM Computer Communications Review*, vol. 35, no. 5, pp. 85–92, Oct. 2005.

[90] NIST, "Secure Hash Standard," National Institute of Standards and Technology (NIST), Federal Information Processing Standards Publication 180-2, Aug. 2002.

[91] J. Nonnenmacher, E. W. Biersack, and D. Towsley, "Parity-Based Loss Recovery for Reliable Multicast Transmission," *IEEE/ACM Transactions on Networking*, vol. 6, no. 4, pp. 349–361, Aug. 1998.

[92] Octoshape Home Page. (2010, Sep.). [Online]. Available: http://www.octoshape.com/

[93] OMA, "DRM Specification," Open Mobile Alliance, OMA-TS-DRM-DRM-V2_1-20081106-A, Nov. 2008.

[94] OpenFEC.org Project Home Page. (2010, Sep.). [Online]. Available: http://openfec.org/

[95] J. Ott, S. Wenger, N. Sato, C. Burmeister, and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)," Internet Engineering Task Force, RFC 4585, Jul. 2006. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4585.txt

[96] V. Pai and A. E. Mohr, "Improving Robustness of Peer-to-Peer Streaming with Incentives," in *Proceedings of the First Workshop on the Economics of Networked Systems*, Jun. 2006, pp. 31–36.

[97] T. Paila, M. Luby, R. Lehtonen, V. Roca, and R. Walsh, "FLUTE - File Delivery over Unidirectional Transport," Internet Engineering Task Force, RFC 3926, Oct. 2004. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3926.txt

[98] T. Paila, R. Walsh, M. Luby, V. Roca, and R. Lehtonen, "FLUTE - File Delivery over Unidirectional Transport," Internet Engineering Task Force, Internet-Draft draft-ietf-rmt-flute-revised-11, Mar. 2010, work in progress. [Online]. Available: http://www.ietf.org/internet-drafts/draft-ietf-rmt-flute-revised-11.txt

[99] B. Pankajakshan and B. J. Parker, "Digital Rights Management for Multicasting Content Distribution," U.S. Patent 7,191,332, Mar. 13, 2007.

[100] C. Perkins, "RTP and the Datagram Congestion Control Protocol (DCCP)," Internet Engineering Task Force, RFC 5762, Apr. 2010. [Online]. Available: http://www.rfc-editor.org/rfc/rfc5762.txt

[101] M. Petit-Huguenin, "Traversal Using Relays around NAT (TURN) Resolution Mechanism," Internet Engineering Task Force, RFC 5928, Aug. 2010. [Online]. Available: http://www.rfc-editor.org/rfc/rfc5928.txt

[102] PLUTO, "Deliverable 2.3 - Simulated Services," The Physical Layer DVB Transmission Optimisation (PLUTO) project, IST-026902/TUT/WP02/RE/P/Del2-3, Aug. 2007. [Online]. Available: http://dea.brunel.ac.uk/pluto/publications/del2-3.pdf

[103] J. Postel, "User Datagram Protocol," Internet Engineering Task Force, RFC 0768, Aug. 1980. [Online]. Available: http://www.rfc-editor.org/rfc/rfc768.txt

[104] ——, "Internet Protocol," Internet Engineering Task Force, RFC 0791, Sep. 1981. [Online]. Available: http://www.rfc-editor.org/rfc/rfc791.txt

[105] ——, "Transmission Control Protocol," Internet Engineering Task Force, RFC 0793, Sep. 1981. [Online]. Available: http://www.rfc-editor.org/rfc/rfc793.txt

[106] J. A. Pouwelse, P. Garbacki, D. H. J. Epema, and H. J. Sips, "The Bittorrent P2P File-Sharing System: Measurements and Analysis," in *Peer-to-Peer Systems IV (Lecture Notes in Computer Science)*. Springer Berlin / Heidelberg, 2005, vol. 3640, pp. 205–216.

[107] PPLive - The Most Popular Net TV in the World. (2010, Sep.). [Online]. Available: http://www.pptv.com/en/

[108] E. Rescorla, "HTTP Over TLS," Internet Engineering Task Force, RFC 2818, May 2000. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2818.txt

[109] J. Rey, D. Leon, A. Miyazaki, V. Varsa, and R. Hakenberg, "RTP Retransmission Payload Format," Internet Engineering Task Force, RFC 4588, Jul. 2006. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4588.txt

[110] R. Rivest, "The MD5 Message-Digest Algorithm," Internet Engineering Task Force, RFC 1321, Apr. 1992. [Online]. Available: http://www.rfc-editor.org/rfc/rfc1321.txt

[111] L. Rizzo, "Effective Erasure Codes for Reliable Computer Communication Protocols," *ACM SIGCOMM Computer Communication Review*, vol. 27, no. 2, pp. 24–36, 1997.

[112] V. Roca, C. Neumann, and D. Furodet, "Low Density Parity Check (LDPC) Staircase and Triangle Forward Error Correction (FEC) Schemes," Internet Engineering Task Force, RFC 5170, Jun. 2008. [Online]. Available: http://www.rfc-editor.org/rfc/rfc5170.txt

[113] P. Rodriguez and E. W. Biersack, "Continuous Multicast Push of Web Documents over the Internet," *IEEE Network Magazine*, vol. 12, no. 2, pp. 18–31, Mar. 1998.

[114] J. Rosenberg, R. Mahy, P. Matthews, and D. Wing, "Session Traversal Utilities for NAT (STUN)," Internet Engineering Task Force, RFC 5389, Oct. 2008. [Online]. Available: http://www.rfc-editor.org/rfc/rfc5389.txt

[115] H. Schulzrinne and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control," Internet Engineering Task Force, RFC 3551, Jul. 2003. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3551.txt

[116] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," Internet Engineering Task Force, RFC 3550, Jul. 2003. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3550.txt

[117] H. Schulzrinne, A. Rao, and R. Lanphier, "Real Time Streaming Protocol (RTSP)," Internet Engineering Task Force, RFC 2326, Apr. 1998. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2326.txt

[118] H. Schulzrinne, A. Rao, R. Lanphier, M. Westerlund, and M. Stiemerling, "Real Time Streaming Protocol 2.0 (RTSP)," Internet Engineering Task Force, Internet-Draft draft-ietf-mmusic-rfc2326bis-24, Jul. 2010, work in progress. [Online]. Available: http://www.ietf.org/internet-drafts/draft-ietf-mmusic-rfc2326bis-24.txt

[119] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the Scalable Video Coding Extension of the H.264/AVC Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1103–1120, Sep. 2007.

[120] P. Shah and J. F. Paris, "Peer-to-Peer Multimedia Streaming Using BitTorrent," in *Proceedings of the 26th IEEE International Performance Computing and Communications Conference (IPCC2007)*, Apr. 2007, pp. 340–347.

[121] Skype - Free Calls, Video Calls and Instant Messaging over the Internet. (2010, Sep.). [Online]. Available: http://www.skype.com/

[122] Sonera - Koti TV (in Finnish). (2010, Sep.). [Online]. Available: http://www.sonera.fi/tv/koti+tv/

[123] SopCast - Free P2P Internet TV. (2010, Sep.). [Online]. Available: http://www.sopcast.org/

[124] K. Sripanidkulchai, A. Ganjam, B. Maggs, and H. Zhang, "The Feasibility of Supporting Large-Scale Live Streaming Applications with Dynamic Application End-Points," in *Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM 2004)*. New York, NY, USA: ACM, Aug. 2004, pp. 107–120.

[125] R. Steinmetz and K. Wehrle, *Peer-to-Peer Systems and Applications (Lecture Notes in Computer Science)*. Springer-Verlag New York, Inc., 2005, vol. 3485.

[126] D. A. Tran, K. A. Hua, and T. Do, "ZIGZAG: An Efficient Peer-to-Peer Scheme for Media Streaming," in *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2003)*, vol. 2, Mar. 2003, pp. 1283–1292.

[127] TVAnts Home Page. (2008, Jan.). [Online]. Available: http://www.tvants.com/

[128] TVU networks - Live TV from around the World. (2010, Sep.). [Online]. Available: http://www.tvunetworks.com/

[129] S. Venaas and T. Chown, "Source Specific Multicast (SSM) with IPv6," in *Proceeding of the 2005 Symposium on Applications and the Internet Workshops (SAINT 2005 Workshops)*, Jan. 2005, pp. 64–67.

[130] R. Walsh, I. Curcio, S. Peltotalo, J. Peltotalo, and H. Mehta, "SDP Descriptors for FLUTE," Internet Engineering Task Force, Internet-Draft draft-mehta-rmt-flute-sdp-06, Jul. 2010, work in progress. [Online]. Available: http://www.ietf.org/internet-drafts/draft-mehta-rmt-flute-sdp-06.txt

[131] M. Wang and B. Li, "Network Coding in Live Peer-to-Peer Streaming," *IEEE Transactions on Multimedia*, vol. 9, no. 8, pp. 1554–1567, Dec. 2007.

[132] M. Watson, "Basic Forward Error Correction (FEC) Schemes," Internet Engineering Task Force, RFC 5445, Mar. 2009. [Online]. Available: http://www.rfc-editor.org/rfc/rfc5445.txt

[133] M. Watson, M. Luby, and L. Vicisano, "Forward Error Correction (FEC) Building Block," Internet Engineering Task Force, RFC 5052, Aug. 2007. [Online]. Available: http://www.rfc-editor.org/rfc/rfc5052.txt

[134] B. Williamson, *Developing IP Multicast Networks*. Cisco Press, 1999.

[135] Windows Media Digital Rights Management. (2010, Sep.). [Online]. Available: http://www.microsoft.com/windows/windowsmedia/drm/default.mspx

[136] Wireshark Packet Analyzer. (2010, Sep.). [Online]. Available: http://www.wireshark.org/

[137] P.-J. Wu, J.-N. Hwang, C.-N. Lee, C.-C. Gau, and H.-H. Kao, "Eliminating Packet Loss Accumulation in Peer-to-Peer Streaming Systems," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, no. 12, pp. 1766–1780, Dec. 2009.

[138] D. Xu, S. S. Kulkarni, C. Rosenberg, and H.-K. Chai, "Analysis of a CDN-P2P Hybrid Architecture for Cost-Effective Streaming Media Distribution," *Multimedia Systems Journal*, vol. 11, no. 4, pp. 383–399, April 2006.

[139] YouTube - Broadcast Yourself. (2010, Sep.). [Online]. Available: http://www.youtube.com/

[140] J. Yu and M. Li, "CBT: A proximity-aware peer clustering system in large-scale BitTorrent-like peer-to-peer networks," *Computer Communications*, vol. 31, no. 3, pp. 591–602, Feb. 2008.

[141] M. Zhang, Q. Zhang, L. Sun, and S. Yang, "Understanding the Power of Pull-Based Streaming Protocol: Can We Do Better?" *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 9, pp. 1678–1694, December 2007.

[142] X. Zhang, J. Liu, L. Boand, and T.-S. P. Yum, "CoolStreaming/DONet: a Data-driven Overlay Network for Peer-to-Peer Live Media Streaming," in *Proceeding of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM2005)*, vol. 3, Mar. 2005, pp. 2102–2111.

[143] H. Zimmermann, “OSI Reference Model – The ISO Model of Architecture for Open Systems Interconnection,” *IEEE Transactions on Communications*, vol. 28, no. 4, pp. 425–432, Apr. 1980.

PUBLICATIONS

PUBLICATION P1

Jani Peltotalo, Sami Peltotalo, Jarmo Harju, and Rod Walsh, "Performance analysis of a file delivery system based on the FLUTE protocol," in *International Journal of Communication Systems*, Volume 20, Issue 6, October 5 2006, pp. 633–659. doi:10.1002/dac.835

PUBLICATION P2

Jani Peltotalo, Sami Peltotalo, Alex Jantunen, Lassi Väätämöinen, Jarmo Harju, Rami Lehtonen, and Rod Walsh, "A Massively Scalable Persistent Content Distribution System," in *Proceedings of the Sixth IASTED International Conference on Communications, Internet, and Information Technology (CIIT 2007)*, Banff, Alberta, Canada, July 2–4 2007, pp. 255–261.

PUBLICATION P3

Jani Peltotalo, Jarmo Harju, Alex Jantunen, Marko Saukko, Lassi Väätämöinen, Igor D. D. Curcio, Imed Bouazizi, and Miska M. Hannuksela, "Peer-to-Peer Streaming Technology Survey," in *Proceeding of the Seventh International Conference on Networking (ICN 2008)*, Cancun, Mexico, April 13–18 2008, pp. 342–350.
doi:10.1109/ICN.2008.86

Seventh International Conference on Networking

Peer-to-Peer Streaming Technology Survey

Jani Peltotalo, Jarmo Harju, Alex Jantunen, Marko Saukko, Lassi Väätämöinen
Tampere University of Technology, Department of Communications Engineering
P.O.Box 553, FIN-33101 Tampere, Finland
Email: forename.surname@tut.fi

Igor Curcio, Imed Bouazizi, Miska Hannuksela
Nokia Research Center
Visiokatu 1, FIN-33720 Tampere, Finland
Email: forename.surname@nokia.com

Abstract

Lately there has been a growing interest in the use of peer-to-peer technologies for deploying large-scale live media streaming systems over the Internet. In this paper we give a brief survey on the peer-to-peer streaming field and have also a closer look on selected applications. In practice, we analyse the selected peer-to-peer streaming systems (Octoshape, SopCast, TVAnts and TVU networks) and see if they are suitable for mobile usage. This paper also shows results from an experimental test carried out by using selected applications with a PC over different network connections (EDGE, UMTS, HSDPA, ADSL and LAN). None of the selected applications are designed to be used in a mobile environment, so there is still a lot of work to do to have optimized systems in the mobile network environment.

1. Introduction

As the amount of media delivered in the Internet seems to be ever-growing and the speeds of end-users' access network connections are getting faster day by day, network capacity continues to be a scarce resource. As opposed to the traditional client/server architecture, some solutions to relieve network congestion have been proposed, different peer-to-peer techniques being the most interesting and popular ones. In a peer-to-peer network, a single host acts as a server and a client simultaneously. Although peer-to-peer techniques do not seem to directly reduce network load compared to client/server approach, it has been observed that network load is distributed more evenly to the whole network when using peer-to-peer techniques. This leads to single links within the network being less congested.

Peer-to-peer streaming is a method for multicasting or broadcasting streaming media, for example audio or video, over the Internet using a peer-to-peer network. It can be seen as a combination of traditional television or radio broadcast type of media delivery over a new kind of delivery medium, the Internet. The aim for these techniques is to allow bandwidth-consuming streaming media to be delivered to a large number of consumers without unnecessary network congestion.

There are special requirements for the access networks and peer-to-peer streaming applications when they are used in the mobile environment. For example delay, jitter and throughput in the access network, used content encoding format, stream bitrate and buffer size affect to the quality of experience and the usability of the application.

Next, the related technologies are discussed in Section 2. Then an introduction to the peer-to-peer streaming concept is given in Section 3. A closer look on selected applications and the results of an experimental test carried out by using these applications are given in Sections 4 and 5. Finally, Section 6 concludes this paper.

2. Related Technologies

In this section we give an overview of the technologies that are commonly used for delivering digital content to a large number of users in the Internet.

2.1. Multicast

Multicast is an elaboration from the simple unicast delivery scheme. While still utilizing the nature of client/server architecture, the amount of data delivered within the network can be remarkably reduced compared to the unicast

978-0-7695-3106-9/08 $25.00 © 2008 IEEE
DOI 10.1109/ICN.2008.86

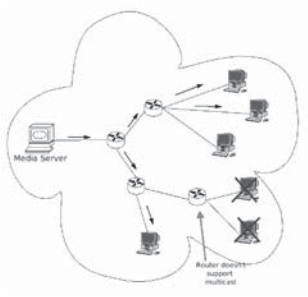

Figure 1: Multicast delivery scheme

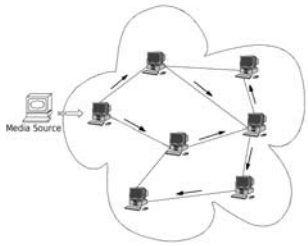

Figure 2: Peer-to-Peer overlay network

delivery. This is achieved by using point-to-multipoint delivery. Multicast delivery scheme is presented in Figure 1.

When using multicast, an IP datagram does not need to be replicated at the server end, but the same datagram is delivered for each receiver with a minimum amount of replication. The receivers must register to "listen" the multicast traffic in order to receive the data delivered by the server. Also, the network infrastructure (i.e., routers) must support multicast traffic to make it possible to forward the data stream to the end-users. As presented in Figure 1, one of the routers does not support forwarding multicast traffic, so the users located in the network behind the router are not capable of receiving the data, even if desired. So the network infrastructure sets a restriction to service availability for the users wishing to receive the service.

2.2. Application Layer Multicast

In application layer multicast IP datagrams are replicated at the end hosts, compared with native multicast delivery where IP datagrams are replicated at the routers. In practice, the end-hosts form an overlay network, and the goal is to construct and maintain an efficient overlay for data transmission. Since application layer multicast protocols possibly send identical packets over the same link (depending on the overlay network topology), they are less efficient compared to native multicast protocols. In contrast there is no need to change routers, so the network infrastructure does not set a restriction to service availability.

2.3. Peer-to-Peer Overlay

A peer-to-peer network utilizes a client/server architecture between number of hosts in a network. That is, each host acts as a server and client simultaneously. The network is depicted in Figure 2 and, as it is obvious, there are no routers in this architecture. This is because a peer-to-peer network is actually an overlay network. This means that the logical connections between hosts, peers, are formed on higher level than the network (IP) level. Typically the peer connections are formed using TCP.

Each host acts as a server and a client and the actual traffic is unicast (point-to-point) in nature. But because the hosts are distributed all over the network, the actual traffic load is distributed more evenly over the whole network. In contrast to an ordinary client/server architecture, where there is one centralized server (or more if the server system is distributed) serving all client hosts, a peer-to-peer network utilizes multiple "servers" distributed over the whole network.

Before being able to transfer data in a peer-to-peer network, a host must first somehow join the overlay network, and the means to do that vary between different peer-to-peer protocols. There is one major advance compared to multicast, though: because peer-to-peer traffic often relies on the existing network infrastructure capable of unicast traffic, there are no (or only a few) users left outside of service from the traffic reachability point of view.

Because of the lack of actual media server in a peer-to-peer network, there must be some way to inject the content for delivery to the network. This is usually done by making the data available on one or more hosts within the network (media source), allowing the content to be delivered to the users. This makes content controlling difficult in the network, when compared to centralized content servers used in multicast and unicast networks.

3. Peer-to-Peer Streaming

Peer-to-peer streaming is a concept for distributing streaming content over a peer-to-peer network. Also the term application layer multicast is sometimes used because application layer connections are used to form the peer connections. The streaming media needs to be injected to the network for delivery, and it is further being delivered

through the whole network to the clients wishing to receive the data. Sometimes the users may not want to receive the data (e.g., a TV channel), but will act as a relay node, also referred to as ”reflector”, to other clients within the network. Reflectors are hosts that pass the streaming data to other hosts without consuming it themselves.

As these networks are peer-to-peer networks, the user wishing to receive (or act as a reflector) needs to join the network before the actual data traffic can occur. After the user has joined the network, there is a varying warm-up time before any data can be consumed. This is because of the initial buffering of data before a stream can be presented in order to ensure seamless viewing or listening of streaming media. The length of the warm-up time depends on the amount of users attending in the network, as well as the users' network capacity and the overall network latency. Also the software used for receiving and playing the stream and stream encoding format affects to the duration of the warm-up time. When a larger reception buffer is used in the software, the warm-up time is longer. A higher quality (low compression ratio and higher bitrate) stream usually takes a longer warm-up time than low bit rate stream of low quality.

3.1. Network Layout

The network layout varies depending on the technique in use. Typical layouts are tree and mesh. In a tree layout network, the stream is divided to several hosts in each node, so that after each node the amount of receivers within the network is multiplied. So, each host in the network acts as a point-to-multipoint server, so that one host receives one stream and delivers it down to several hosts. In a mesh network, each node is connected to several other hosts, and each can receive and send out multiple streams.

On a tree layout network there exists the single point of failure type of problem. When considering strictly a tree layout network, every node of the tree (excluding leaf nodes) is a root node having one or more child nodes. A stream is always passed through the root node to its children. If any of the root nodes happens to fail, the whole network originating from the failed root node fails to receive the stream. The worst case scenario is that the primary root for the whole tree fails, leading to denial of service for the whole network.

Such single point of failure issues can be avoided, or at least reduced in a combined tree-mesh or a mesh layout network. In these kind of networks all or at least some of the peers (nodes) have more than one connection to other nodes, so that instead of just passing the stream from root to children, also the children to root direction is used. This allows other nodes to receive the stream when a single (root) node fails. Thus compared to tree layout, mesh layout requires more complicated routing algorithms or request mechanisms between peers due to the increased number of peer connections.

3.2. Push and Pull Methods

One common feature shared by some peer-to-peer streaming systems is that they are push-based systems. This means that after a peer has received data (the stream) it sends it on to other peers in the network, without explicit requests for data from other peers. The forwarding decision is based on some predetermined routing algorithm, and the same algorithm is globally used over the whole network. This also leads to the network layout to be somewhat rigid, at least to some extent, because it is determined by the routing algorithm.

The problem with push-based systems is that they are poor in recovering from transmission losses, caused by the lack of requests for data. For example, if a peer connection is broken between two peers, a sending peer fails to forward the data to the receiving peer across this broken connection. This leads to the receiving peer never receiving data, because of the broken connection, thus experiencing a corrupted stream. Another problem in a push-based network is the amount of duplicate data. Because of the routing algorithm used for ”blindly” forwarding (pushing) the data, it may well be that one or more peers may be sending the same packets to a common destination host. [7] This can be avoided using requests to get the desired packets from the sending peers, leading to pull-based system.

In a pull-based system, peers wishing to receive the stream request the missing packets from other peers. That is, a peer wishing to receive a packet from other peers must request it prior to receiving. After receiving a packet, peer must notify other peers about the packet it received in order to pass the stream along in the network, thus enabling other peers to request the data.

However, if for some reason a packet is not received by a peer, it may request it from one or more peers announcing to have that packet. This results in better resilience against packet loss in reception, because in case of a failure the receiving peer can redirect request packets to another peer having the desired data. Also, when using a request based method, there is no need for using predetermined routing algorithms as in the push-based approach. [7] The amount of duplicate data sent within the network is reduced, because the requests from the peer wishing to receive a packet may only be addressed to one sender who provides the packet to the receiver.

An obvious weakness with the pull-based method is the issue of dealing with free-riders. A free-riding peer is only requesting and receiving packets from other peers, without uploading anything to others. It is obvious that this affects to the performance of the network, because free-riders do

not send requested packets to other peers.

In a pull-based network the layout may well be more ad hoc in nature, because there might not be any predetermined way to form peer-relations, but the requests and announcements sent within the network define peer-relations.

3.3. Multiple Stream Approach

To deal with, e.g., the free-riding problem apparent a in single stream based delivery utilizing peer-to-peer streaming networks, there is an alternative approach to deliver streaming data: using multiple streams. In this approach a single stream (full stream) is divided into several substreams, also referred to as "descriptions" [5]. Each substream is then forwarded separately in the network. A peer receiving all of the substreams can reconstruct the full-quality original stream from the sub-streams received, and therefore enjoys the best quality.

Using multiple streams in a peer-to-peer streaming network is a potential way for implementing an incentive mechanism [8] in such a network. The incentive mechanism can be considered similar to "scoring" in traditional peer-to-peer file sharing networks, where the uploading peers are scored by the amount of data they upload. The more scores a peer has, the more privileged are the peer's download opportunities. In a streaming network, a peer uploading more streams has better "scores", thus being entitled to download more substreams and getting a higher quality stream itself. For example, for every uploaded stream, a peer is entitled to download one stream, thus enabling fairness among the peers in the network.

In addition to free-riding, the multiple stream approach helps to relieve the churning phenomenon also. If a peer sending out one substream leaves the network, the overall stream quality does not collapse remarkably, because satisfactory quality of the overall stream can be achieved from the substreams that still exist. In churning, multiple (probably hundreds) peers rapidly attach and detach to the network within a short period of time, making substreams rapidly available and unavailable. If we compare the multiple stream approach to the single stream approach in a churning, we avoid the following problematic case. Suppose that a peer sending out full stream leaves the network. If it is the only provider for that stream, the reception of the stream will be cut off from all the peers listening the stream sent by the detached peer. Hence, large amount of peers joining to and departing from the network in a single stream system may cause full loss of stream at times.

3.4. Mobile Aspects

There are special requirements for the access networks and peer-to-peer streaming applications when they are used in a mobile network environment. Delay in the access network has an effect on the buffering period at the beginning of the stream. Furthermore, the role of the application buffer size is much more important. If it is big enough, the effects of delay and jitter can be ignored. The access network throughput limits the quality of the stream: the bigger the used stream bitrate is, the bigger the throughput should be. If the used stream bitrate is too high then it is not possible to use the application over mobile network at all. Currently used content encoding formats do not have much effect, but when optimized solutions for mobile devices will exist then CPU capabilities might exclude some content encoding formats.

All these aspects affect to the quality of experince and the usability of the peer-to-peer streaming application and should be taken into account in a mobile network environment.

4. Closer Look on Selected Applications

In this section a closer look on selected applications, Octoshape [6], SopCast [9], TVAnts [10] and TVU networks [11], is given. These were selected from currently existing software packages, summarized in Table 4 (at the end), after a preliminary study taking into account the aspects defined in Section 3.4. Also MaxTV [3] is introduced in Subsection 4.5, because of its relationship to the selected applications. However, no measurements were done with it.

4.1. Octoshape

Octoshape is a 2003 founded company that offers commercial live broadcasting services. They offer free trial only to companies and organizations and their offer is not applicable for home users [6]. This ensures that the content of their system, even if trial content, will always have decent background.

Because of the commercial nature, the content is offered mostly by companies, e.g., TV and radio stations. The bitrate of the streams vary from 32 kbps for audio streams to 800 kbps for video streams. Even though there is a broadcasting fee, the watching is free of charge and no references for pay television is mentioned [6]. There is, however, at least one channel that has connection restrictions and the Octoshape client warns that "This stream cannot be viewed in your country.".

Octoshape has also broadcasted some big events like the Eurovision Song Contest in 2006 and 2007. They also broadcasted the Tour de France in 2007 in cooperation with the Belgian broadcaster VRT with quality as high as 1.5 Mbps [6].

Octoshape works on Windows (2000 and later versions), Linux and Max OS X, of which the two latter ones are in

beta stage. The system requirements are such that almost any web browser and video player can be used with Octoshape. The client application is quite simple and includes only the connection to the delivery network. The content is searched with a web browser and the streams are played with a third party video player. Neither of them is integrated to the Octoshape client itself.

4.2. SopCast

SopCast is a free peer-to-peer streaming application that can be used for both live and Video on Demand (VoD) services. It is a closed source and is based on *Sop* technology. The SopCast Team started to work on this project in December 2004 and has gained considerable popularity especially in China.

SopCast has its own client and server software built on the Windows operating system. The Software is divided into three different components: SopPlayer, SopServer and WebPlayer. Although they have built their UI in which they embedded everything needed for viewing streams, the software needs either the Windows Media Player 9 (or above), or the latest RealPlayer. Separate media player applications, e.g., VideoLan Client or MPlayer, can also be used for stream playback. In addition to Windows, there is also a console-based software built for Linux platforms which have GUI provided by third party.

To use SopCast, users are not forced to register, and the software can easily be downloaded from the webpage. However, registration is required if a user wishes to broadcast his own media stream. A channel may either be public, and shown to all SopCast users, or private so that your stream can be reached through a link, e.g., after advertising the stream on your own webpage.

Most of the content in SopCast is related to sports, but there are also channels in entertainment and music categories. Channels in SopCast are founded by home users and no TV company partners are mentioned in the home pages.

4.3. TVAnts

TVAnts is a freeware application developed by the Zhejiang University (software copyrighted in 2005). TVAnts provides live broadcasting functionality to everyone with Internet connection.

TVAnts is based on a tracker system where content information is gathered from different trackers and listed in the TVAnts client. All the needed features like content search and watching are integrated to the TVAnts client and no third party softwares are needed to run by the user during the software usage. Even though TVAnts has a lot of users, the client is only available for the Windows operating system. The contents of the channel list is a mixture of Japanese and English languages. Also the channel guides and advertisements are mostly in Japanese.

The content varies a lot and used channel bitrates are between 10 kbps and 900 kbps where the typical value beeing between 300 kbps and 500 kbps. The most popular streams are ”sport broadcasts”, e.g., football, and at times the audience goes over 500 simultaneous users on the most popular ones.

TVAnts offers much more specific information about the network statistics than the other competitors reviewed in this paper, most probably because of the university background. Users may, e.g., watch how the buffer is filled and which parts are available, as well as upload and download statistics, and CPU usage of the different software components (e.g., tracker, transmitter, media player, etc.) are monitored.

4.4. TVU networks

TVU networks was founded in 2005 and it is headquartered in Mountain View, California with Asia Pacific offices also in Changhai, China. TVU offers live broadcast services for home users and companies based on their own technology. Amateur broadcasting and viewing the streams are free of charge. However, for professional broadcasters TVU offers broadcast hardware and services [11].

TVU has set the goal to be a global TV broadcaster and has also some partners, e.g., Black Belt TV, GOD TV, ETTV, The Auto Channel, WCSN, which offer content 24/7 with reasonable quality. From these partners at least WCSN provides currently also one premium channel. In addition to this there are a lot of channels founded by home users with varying quality.

Typical channel bandwidth is between 280 kbps and 400 kbps. There is no limit in the quality, but when broadcasting with higher quality more upload capacity is needed from the broadcaster and clients in order to keep the streams playable.

Currently TVU offers players for Windows (2000, XP and Vista with WMP 9 or above) and broadcasting software for Windows and Linux. The bandwidth requirement for the player is 300 kbps. The system requirements for boadcasting are very low compared to modern computers. The minimum bandwidth requirement for broadcasting is two times the video quality, but in order to ensure good user experience, a connection with around ten times the video quality is recommended. The latest versions of the TVUBroadcaster and TVUPlayer softwares are beta versions, however the Windows client worked very well during our tests [11].

4.5. MaxTV

A system related to the selected technologies is MaxTV [3]. This is capable of playing streams from all of the selected systems via plugins that provides connection to the specific P2P network. Because MaxTV is not the original player of these streams, it was not tested against other systems.

Due to the large number of different P2P streaming solutions in the market, MaxTV is in very good position because it can be extended to support multiple techniques simultaneously. This makes the end users' life easier; there is no need to download and install multiple applications, and the change between channels broadcasted in different networks is managed using only one application.

5. Experimental Tests

In this section, results for experimental tests carried out by using Octoshape, SopCast, TVAnts and TVU networks TVUPlayer with a PC (Intel Core2Duo 6300, 2GB DDR2, running Windows XP SP2) over different network connections (EDGE, UMTS, HSDPA, ADSL and LAN) are presented. We used the latest version of the applications that were available at 9th of January 2008.

Windows XP SP2 is limiting incomplete outbound TCP connection attempts to 10 [4], which slows down the connection establishment to the P2P network. In Windows XP SP1 there was no limitations like this. There are unofficial ways to overcome this limitation, however none of them was used in our tests.

A Nokia N95 was used as a modem for the EDGE connection, and a Nokia DKE-2 cable was used between the PC and the mobile phone. For the UMTS connection we used a Nokia 6680 as a modem with a Nokia CA-53 cable that connected the mobile phone with the PC. With HSDPA, a Nokia N95 was used as a modem and the connectivity to the PC was established through the Nokia DKE-2 cable. The Ericsson HM410dp was used as an ADSL modem in the tests, and it was connected to the PC Gigabit Ethernet controller with an RJ-45 cable.

5.1. Connection Characteristics

Table 1 shows the measured downlink and uplink throughput and the average Round Trip Time (RTT) values for each network connection. The downlink throughput values (DL) are measured by downloading 10 MB file from ftp://ftp.funet.fi/dev/ and using the Wireshark network protocol analyzer [12] to calculate the value. The uplink throughput values (UL) are measured by uploading 10 MB file to one of our university server and using Wireshark to calculate the value. The average RTT values (from 100 measurements) to http://www.ficix.fi/ are measured to get the access network delay. FICIX, the Finnish Communication and Internet Exchange association, is the biggest Internet exchange point in Finland.

Table 1: Network characteristics

	Connection	Throughput (DL/UL)	RTT
Mobile	EDGE	154 kbps / 77 kbps	623 ms
	UMTS	351 kbps / 123 kbps	135 ms
	HSDPA	888 kbps / 347 kbps	88 ms
Leased line	ADSL	1 Mbps / 512 kbps	48 ms
	LAN	100 Mbps / 100 Mbps	5 ms

There were no firewalls to FICIX with leased line connections, but with mobile connections the firewall setup is operator dependent. In the tests, the client software was restarted between the tests, and when starting, the client's initial downloading (webpages, channel list, etc.) was allowed to complete before pressing the play button.

With SopCast and TVAnts it is possible to get information about the number of peers in each channel. In our tests we selected from SopCast and TVAnts only channels that offered a good channel and a reasonably large user population. With TVU networks we selected channels provided by partner companies to ensure that we could get as large user population as possible, because the number of peers in the channels founded by home users is usually quite low. With Octoshape the content is offered mostly by companies and it has good quality, so we selected channels with bitrates closiest to the bitrates used with SopCast, TVAnts and TVU network.

Measurements for one test case, e.g., $\sim$350 kbps stream with SopCast, were conducted using several channels during several days to exclude defects induced by temporary good peers.

5.2. Results

Table 2 shows bootstrap times for different stream qualities with different network connections. The bootstrap time is the time (average from 5 to 20 tests) from pressing the play button to good user experience. From the table it is evident that the boostrap time is dependent on the delay and throughput of the access network as was pointed out in the Section 3.4 also. Bootstrap times with Octoshape are remarkably lower than with other technologies. Th reason for this might be the commercial nature of Octoshape. During the tests we noticed that Octoshape client gets most of the data from one peer, which might be provided by the company. Still from time to time the upload rate of our client was quite high, so there is also peer-to-peer delivery between normal peers.

Table 2: Bootstrap times for different stream qualities with different network connections

Connection	Octoshape			SopCast			TVAnts			TVU networks		
	∼ 100 kbps	∼ 400 kbps	∼ 600 kbps	∼ 64 kbps	∼ 350 kbps	∼ 600 kbps	∼ 100 kbps	∼ 350 kbps	∼ 600 kbps	∼ 100 kbps	∼ 350 kbps	∼ 500 kbps
EDGE	14s [1]	- [2]	- [2]	59s	- [2]	- [2]	- [2]	- [2]	- [2]	105s [1]	- [2]	- [2]
UMTS	8s	- [2]	- [2]	24s	- [2]	- [2]	87s	- [2]	- [2]	47s	- [2]	- [2]
HSDPA	6s	13s	17s	14s	65s	74s	59s	71s	87s	37s	51s	58s
ADSL	4s	10s	11s	14s	62s	69s	62s	72s	80s	31s	56s	62s
LAN	4s	10s	5s	13s	54s	36s	59s	69s	77s	22s	26s	28s

From the home pages of the tested applications it is possible to find information about buffering times, for TVU networks 5-30s and SopCast 30-90s (Octoshape and TVAnts do not publish buffering times). These are somewhat in line with our bootstrap time measurements.

For TVU networks and TVAnts the average stream buffer size was around 45s, for SopCast it was around 47s, and for Octoshape around 32s. These values were measured by disconnecting from the access network and measuring how long the stream continued to play without noticeable interrupts. So our measured buffer size was the time period from the disconnection to the noticeable interrupts. There might be data in the buffer also after this time period, but, e.g., TVAnts stopped to play the stream when the first discontinuity in the buffer occured.

From these values we can conclude that after the initial buffering (boostrap time), delay and jitter in the access network do not have much effect to the user experience, because the big buffer size will smooth the variation between packet arrival times.

Table 3 shows how the quality of the selected applications was seen by our research group. In the table the stream bitrates are categorized as following: Low is up to 150 kbps, Medium is from 300 kbps to 400 kbps, and High is from 500 kbps to 600 kbps. Satisfactory means that the stream was watchable, but from time to time there were some interrupts or buffering that lowered the user experience. Good means that the stream played as expected without errors, but the experience was lowered because of the bootstrap time. Very good means that in addition to the good signal quality, the bootstrap time was low.

We also tested how these technologies worked when the connection is changed from one to another, e.g., from UMTS to ASDL. If the roaming is successful, the user can continue playing the stream all the time without interrupts, which is obvious benefit for mobile usage. From the selected technologies Octoshape and SopCast were able to recover from the connection change quite easily and with the shortest interrupt, but they required the new connection to be available almost instantly when the originating connection was disconnected. TVAnts was also few times able to recover from connection exchange, but usually with noticable interrupts and software restart would be much better choice.

[1] Sometimes the stream works with this connection and sometimes it does not. It is not recommended to use this connection with this stream.

[2] Connection is not suitable to be used with this stream.

Table 3: Quality of Experience

Stream Bitrate	Octoshape	SopCast	TVAnts	TVU networks
Low	Very Good	Satisfactory	Satisfactory	Good
Medium	Very Good	Satisfactory	Satisfactory	Good
High	Very Good	Satisfactory	Satisfactory	Good

With TVUPlayer we noticed that it remained online in the background (the icon stays in the notification area) when it is closed normally. This allows TVU to collect idle users to share their upload bandwidth to increase the stream availability and also allows a faster resume of the stream playback, but as a disadvantage this could also just consume bandwidth if the user does not have more upload bandwidth than the download consumes.

As a conclusion, our experimental tests show that all applications are suitable for mobile usage with high-throughput mobile networks, but still it is clear that they are designed to be used with leased line connections.

6. Conclusion

At the moment peer-to-peer streaming is securing its position among users and the implementations are more or less unfinished or oriented to a small group of broadcasters. Compared to traditional unicast media transfer, peer-to-peer streaming offers a lot of improvements concerning the bandwidth usage and server requirements. Although the improvements are great and help small organizations to build, e.g., Internet radio to a large audience rather easily, people are probably unaware of the opportunities.

Compared to YouTube [13] and other similar techniques, peer-to-peer streaming solutions are still far from being ma-

ture. A lot of work needs to be done for example on usability, in order to gain massive public support. People do not always want to, or even cannot, download the client software or share their bandwidth for security reasons, which prevents effective spreading of these techniques.

For the average user, YouTube allows easy-to-access and easy-to-use services in order to spread user content and to share experiences in a community-like environment. But if the intention is that of professionally broadcasting high quality content over the Internet to a large-scale audience, then peer-to-peer streaming is the best choice.

The currently existing peer-to-peer streaming applications are not implemented with a mobile network environment philosophy but, as this paper shows, some of them are already suitable to be used in the mobile environment. Still there are many issues to be solved before optimized solutions for mobile devices will be available. One way to enhance the mobile usage with current applications is to implement support for multiple stream approach (see Section 3.3), e.g., with Multiple Description Coding (MDC) [1] or Advanced Video Coding (AVC) [2]. In this way clients with low throughput access network connections are also able to play the stream and the overall quality of the stream increases.

References

[1] V. K. Goyal. Multiple Description Coding: Compression Meets the Network. *IEEE Signal Processing Magazine*, 18(5):74–94, Sept. 2001.

[2] ISO/IEC 14496-10:2005. Information technology – Coding of audio-visual objects – Part 10: Advanced Video Coding, 2005.

[3] MaxTV. Homepage, 2007. Available http://www.max-tv.be/. Accessed 8 January 2008.

[4] Microsoft Corporation. Part 2: Network Protection Technologies, 2004. Available http://technet.microsoft.com/en-us/library/bb457156.aspx#EHAA. Accessed 8 January 2008.

[5] J. Mol, D. Epema, and H. Sips. The Orchard Algorithm: P2P Multicasting without Free-Riding. In *Proceedings of the Sixth IEEE International Conference on Peer-to-Peer Computing*, Sept. 2006.

[6] Octoshape. Homepage, 2007. Available http://www.octoshape.com/. Accessed 3 January 2008.

[7] V. Pai, K. Kumar, K. Tamilmani, V. Sambamurthy, and A. E. Mohr. Chainsaw: Eliminating Trees from Overlay Multicast. In *Proceedings of the Fourth International Workshop on Peer-to-Peer Systems*, Feb. 2005.

[8] V. Pai and A. E. Mohr. Improving Robustness of Peer-to-Peer Streaming with Incentives. In *Proceedings of the First Workshop on the Economics of Networked Systems*, June 2006.

[9] SopCast. Homepage, 2006. Available http://www.sopcast.org/. Accessed 2 January 2008.

[10] TVAnts. Homepage, 2007. Available http://www.tvants.com/. Accessed 3 January 2008.

[11] TVU Networks Corporation. Homepage, 2007. Available http://www.tvunetworks.com/. Accessed 2 January 2008.

[12] Wireshark. Homepage, 2008. Available http://www.wireshark.org/. Accessed 15 January 2008.

[13] YouTube. Homepage, 2007. Available http://www.youtube.com/. Accessed 2 January 2008.

A. Summary of Existing Peer-to-Peer Streaming Technologies

Table 4 summarises currently existing peer-to-peer streaming technologies which have been active (at least have a working home page) on January 2008. Technologies are divided into three categories (freeware, commercial, and Far East). The last gategory exists since for those technologies it was quite difficult to obtain enough information for a different classification, because there were no English versions of the web pages available.

Table 4: Summary of existing peer-to-peer streaming technologies

	Software	Home page	Live	VoD
Freeware	ACTLab TV	http://www.actlab.tv/		X
	CloneCast	http://clonecast.free.fr/	X	
	Coolstreaming Mediacenter	http://www.coolstreaming.it/en/	X	
	Cybersky-TV	http://www.cybertelly.com/	X	
	End System Multicast (ESM)	http://esm.cs.cmu.edu/	X	
	Freecast	http://www.freecast.org/	X	
	Nodezilla	http://www.nodezilla.net/	X	
	P2P-Radio	http://p2p-radio.sourceforge.net/	X	
	P2PLive	http://www.p2plive.org/	X	
	Peer-to-Group Media Broadcast	http://www.alphaworks.ibm.com/tech/p2g/	-	-
	PeerCast	http://www.peercast.org/	X	
	PPLive	http://www.pplive.com/	X	X
	Seecast (Google Code)	http://code.google.com/p/seecast/	-	-
	SopCast	http://www.sopcast.com/	X	X
	Stream-2-stream	http://s2s.sourceforge.net/	X	
	Trevbus	http://www.trevbus.org/	X	
	Tribler Streaming	http://tribler.org/test_streaming/	X	
	TVAnts	http://www.tvants.com/	X	
	TVU networks & Viidoo	http://www.tvunetworks.com/	X	
	VidTorrent	http://viral.media.mit.edu/index.php?page=vidtorrent	-	-
Commercial	Abacast	http://www.abacast.com/	X	
	AllCast	http://www.allcast.com/	X	
	BitTorrent	http://www.bittorrent.com/	X	X
	CCIPTV	http://en.cciptv.com/	X	
	Grid Networks	http://www.gridnetworks.com/	X	
	Itiva	http://www.itiva.com/	X	X
	JoostTM	http://www.joost.com/		X
	MaxTV	http://www.max-tv.be/?lng=en	X	
	Mediazone	http://www.mediazone.com/	X	X
	NeoKast	http://www.neokast.com/	X	
	Network Foundation Technologies (NFT)	http://www.nft-tv.com/	X	
	Octoshape	http://www.octoshape.com/	X	
	PeerStream	http://www.peerstream.net/	X	
	RawFlow	http://www.rawflow.com/	X	
	StreamAudio / ChainCast	http://www.streamaudio.com/	X	
	StreamerOne	http://www.streamerone.com/	X	
	Streamer P2P	http://www.streamerp2p.com/	X	
	Swarmcast	http://www.swarmcast.com/	X	X
	SyncCast	http://www.synccast.com/	X	X
	Veoh	http://www.veoh.com/	X	X
	Vuze	http://www.vuze.com/		X
	Zattoo	http://zattoo.com/	X	
Far East	Afreeca & Pdbox	http://afreeca.pdbox.co.kr/	X	
	Gridmedia	http://www.gridmedia.com.cn/	X	
	Mysee	http://www.mysee.com/	X	
	Pcast	http://itv.mop.com/	X	
	PPStream	http://ppstream.com/	X	
	PPMate	http://www.ppmate.com/	X	
	PPStream	http://ppstream.com/	X	
	SynaCast	http://www.synacast.com/	X	
	QQLive	http://tv.qq.com/	X	
	Roxbeam	http://www.roxbeam.com/	X	X
	ShareCast	http://www.scast.tv/scast/	X	
	Uusee	http://www.uusee.com/	X	
	Vakaka	http://vakaka.com/		X

PUBLICATION P4

Jani Peltotalo, Jarmo Harju, Marko Saukko, Lassi Väätämöinen, Igor D. D. Curcio, and Imed Bouazizi, "Personal Mobile Broadcasting based on the 3GPP MBMS System," in *Proceedings of the 6th International Conference on Advances in Mobile Computing & Multimedia (MoMM2008)*, Linz, Austria, November 24–26 2008, pp. 156–162. doi:10.1145/1497185.1497219

PUBLICATION P5

Jani Peltotalo, Jarmo Harju, and Miska M. Hannuksela, "Reliable, Server-Friendly and Bandwidth-Efficient File Delivery System using FLUTE Server File Format," in *Proceedings of the IEEE International Symposium on Broadband Multimedia Systems and Broadcasting 2009 (BMSB2009)*, Bilbao, Spain, May 13–15 2009, pp. 1–6. doi:10.1109/ISBMSB.2009.5133753

Reliable, Server-Friendly and Bandwidth-Efficient File Delivery System using FLUTE Server File Format

Jani Peltotalo*, Jarmo Harju* and Miska M. Hannuksela†

* Tampere University of Technology, Department of Communications Engineering
P.O.Box 553, FI-33101 Tampere, Finland
Email: forename.surname@tut.fi

† Nokia Research Center
P.O.Box 1000, FI-33721 Tampere, Finland
Email: forename.surname@nokia.com

***Abstract*—The use of Forward Error Correction (FEC) codes is a classical solution to improve the reliability of multicast and broadcast transmissions in a packet erasure channel. However, FEC encoding on-the-fly increases the load of the server and it may decrease the overall performance of the file delivery system. This paper presents a reliable, server-friendly and bandwidth-efficient file delivery system using the server file format for File Delivery over Unidirectional Transport (FLUTE) protocol. The FLUTE server file format enables storage of pre-composed source symbols and pre-calculated FEC symbols into a media container file, so there is no need to source symbol construction and FEC encoding on-the-fly. Additionally, the actual transmission is controlled by file delivery hint tracks containing cookbook instructions that ease the encapsulation of source and FEC symbols into a transmittable packet stream.**

***Index Terms*—File Delivery System, FLUTE, ISO Base Media File Format, FLUTE Server File Format**

I. Introduction

The media container file format is an important element in the chain of multimedia content creation, manipulation, transmission and consumption. The file format comprises means of organizing the generated bit stream in such way that it can be accessed for local decoding and playback, transferred as a file, or streamed, all utilizing a variety of storage and transport architectures. The ISO base media file format [1] is a base format for many different media file formats. For example MP4 file format (ISO/IEC 14496-14 [2]), AVC file format (ISO/IEC 14496-15 [3]) and 3GPP file format (3GPP TS 26.244 [4]) are based on the ISO base media file format.

The FLUTE server file format consists of features that are a part of Edition 3 of the ISO base media file format [1] and Amendment 1 for it [5]. Files intended for the delivery are partitioned into several source blocks, and each source block is then stored as a file reservoir item in a media container file. For each source block additional Forward Error Correction (FEC) symbols can be pre-computed and stored as a FEC reservoir item. The actual transmission is controlled by File Delivery (FD) hint tracks containing instructions that ease the encapsulation of source and FEC symbols into packets.

Earlier studies have shown that the use of FEC is a good option to improve the reliability of a file delivery system in a packet erasure channel [6]. As is shown in [7], the overhead in a data carousel [6] caused by the guarantee of reliability is much lower when FEC is added to the data carousel. However, the load of the file delivery server might increase if the FEC encoding is done on-the-fly. This extra server load can be avoided and the overall system performance increased by using the FLUTE server file format.

The structure of the remainder of this paper is as follows. Next, short overviews of the FLUTE protocol, ISO base media file format and FLUTE server file format are given in Sections II−V. Then the architecture of a file delivery system using the FLUTE server file format is described in Section VI. After that performance analysis of the system is given in Section VII. Finally, Section VIII concludes this paper.

II. FLUTE Protocol

File Delivery over Unidirectional Transport (FLUTE) [8] has been widely adopted as the file delivery protocol for multicast and broadcast applications. FLUTE is based on the Asynchronous Layered Coding (ALC) protocol [9], and the Layered Coding Transport (LCT) protocol [10]. LCT provides transport level support for reliable content delivery and stream delivery protocols. ALC is a protocol instantiation of the LCT building block, and it serves as a base protocol for massively scalable reliable multicast distribution of arbitrary binary objects. FLUTE builds on top of ALC/LCT and defines a protocol for unidirectional delivery of files. FLUTE therefore inherits all of the functionalities defined in the ALC and LCT protocols.

LCT defines the notion of LCT channels to allow massive scalability, which has been designed based on the receiver-driven layered multicast principle, where receivers are re-

sponsible of implementing an appropriate congestion control algorithm based on the adding and removing of layers of the delivered data. The sender sends the data into different layers, with each being addressed to a different multicast group.

One or multiple LCT channels may be used for the delivery of the files of a FLUTE session. A great flexibility is given to the FLUTE sender with regard to how the data is partitioned among the LCT channels. A common use case is to send the same content on all different LCT channels but coded at different bit rates. Additionally, the FLUTE sender may act intelligently to enable receivers to acquire all files of the FLUTE session by joining all channels for a shorter time than is normally required with one channel. In such a case, the data sent over each channel complements the data of other channels.

FLUTE defines a File Delivery Table (FDT), which carries metadata associated with the files delivered in the FLUTE session, and provides mechanisms for in-band delivery and updates of the FDT.

III. Forward Error Correction

FEC codes can be divided into systematic and non-systematic codes. With a systematic FEC code, such as Reed-Solomon FEC [11], the first portion of a FEC encoding block consists of source symbols, i.e., the original content items for the given block, while the remaining symbols for the block consist of FEC symbols generated by a FEC encoder. In this case, the receiver must receive any set of encoding symbols equal in number to the number of source symbols.

When a non-systematic FEC code is used, all symbols for the block consist of symbols generated by the FEC encoder. In this case, the receiver must receive a sufficient number of symbols to reconstruct the original user data for the given block via FEC decoder. There exists a couple of alternative systematic FEC encoding schemes, Raptor FEC [12] (named MBMS FEC in 3GPP TS 26.346 [13]) being one of the most widely used among different standardization organizations.

With Raptor FEC, the operation of a FEC encoder is divided into several steps. First, the source file is divided into $Z \geq 1$ source blocks and FEC encoding is applied independently to each source block. Next, each source block is divided into $N \geq 1$ contiguous sub-blocks. After that, each sub-block is divided into K sub-symbols and the mth symbol of a source block consists of the concatenation of mth sub-symbol from each of the sub-blocks. It should be noted that when $N > 1$, then a source symbol is not a contiguous portion of the source file. This happens when the source file size is bigger than the target sub-block size (the recommended value of which is 256 KB). Finally, the FEC encoder generates a desired number of FEC symbols for each source block that consists of K source symbols.

It is not necessary for the receiver to know the total number of FEC symbols (per source block) with the Raptor FEC. The receiver receives some set of encoding symbols, slightly more in number than the number of source symbols, and feeds those to an FEC decoder. From these encoding symbols the FEC decoder generates K source symbols, divides each source symbol to N sub-symbols, and composes N sub-blocks which can be concatenated to re-establish the original source block.

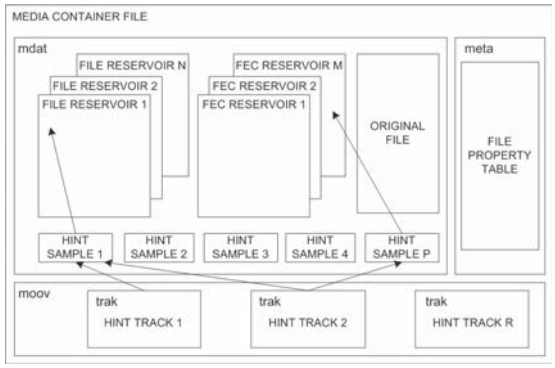

Fig. 1. An example media container file

IV. ISO Base Media File Format

The basic building block in the ISO base media file format is called a box. Each box has a header and a payload. The box header indicates the type of the box and the size of the box in terms of bytes. A box may enclose other boxes, and the ISO base media file format specifies which box types are allowed within a certain box. Furthermore, some boxes are mandatorily present in each file, while others are optional. Moreover, for some box types, it is allowed to have more than one box present in a file. So, the ISO base media file format actually specifies a hierarchical structure of boxes.

According to the ISO base media file format, an ISO base media file consists of metadata and media data that are enclosed in separate boxes, the movie box ($moov$) and the media data box ($mdat$), respectively. The movie box may contain one or more tracks, and each track resides in one $trak$ box. A media track refers to samples formatted according to a media compression format and its encapsulation to the ISO base media file format. A hint track refers to hint samples, containing cookbook instructions for constructing packets for transmission over an indicated communication protocol.

It should be noted that the ISO base media file format does not limit a presentation to be contained in a single file; it may be contained in several files. One file contains the metadata for the whole presentation. This file may also contain all the media data, whereupon the presentation is self-contained. The other files, if used, are used to contain media data and are not required to be formatted according to the ISO base media file format. The ISO base media file format concerns the structure of the presentation file only.

In addition to timed tracks, the ISO base media file can contain any non-timed binary items. The $meta$ box may list and characterize any number of binary items that can be referred to and each one of them can be associated with a

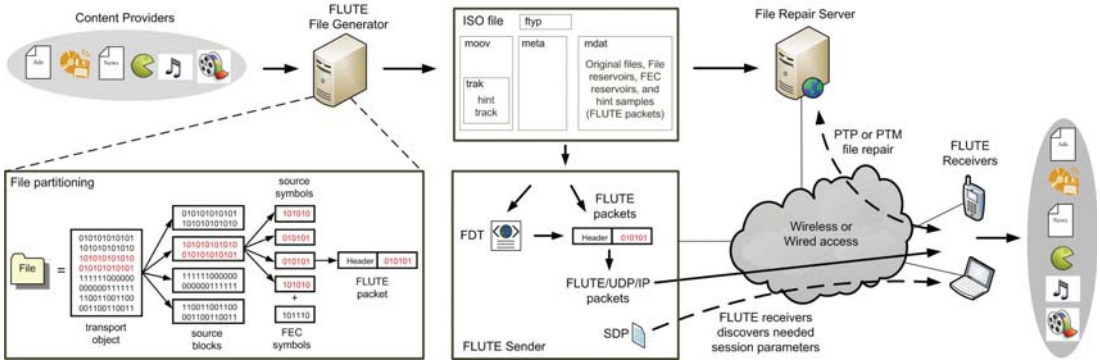

Fig. 2. System Architecture

file name and unique item identifier.

V. FLUTE Server File Format

Fig. 1 shows an example media container file with one source file. In this example, each source block consists of more than one sub-block, so a source symbol is not a contiguous portion of the source file. Consequently, it is not possible to include source symbols by reference to the original source file.Instead, the media container file contains three file reservoirs labeled File reservoir 1, 2, and N, and an equal amount of FEC reservoirs labeled FEC reservoir 1, 2, and M. In a general case, any number N file reservoirs and M FEC reservoirs can be stored in a media container file, and typically N equals to M. When the media container file is formatted according to the ISO base media file format, each file reservoir and FEC reservoir is a binary item of the ISO base media file format.

The file property table can be formatted similarly to the FD Item Information Box of the ISO base media file format. It contains an association meta data to identify items that are file reservoirs and items that are FEC reservoirs. In addition, the association meta data logically links each respective pair of a file reservoir and FEC reservoir with each other, i.e., the source symbols of a source blocks and the FEC symbols derived from the source block. In practice, the association meta data can be a loop or a table of partition entries as described subsequently.

The media container file may additionally comprise any number of hint tracks for instructing in deriving packets from file and FEC reservoirs for file delivery. The hint tracks can be formatted according to the FD hint tracks of the ISO base media file format. File and FEC reservoirs can be used independently of FD hint tracks and vice versa. The reservoirs aid the design of hint tracks and allow alternative hint tracks, for example with different FEC overheads, to re-use the same FEC symbols. They also provide means to access source symbols and additional FEC symbols independently for post-delivery repair, which may be performed over FLUTE or out-of-band via another protocol. In order to reduce complexity when a server follows hint track instructions, hint samples refer directly to the data ranges of the items to be copied into the hint samples.

The support for file delivery is designed to optimize the server transmission process by enabling FLUTE servers to follow simple instructions. It is enough to follow one pre-defined sequence of instructions per channel in order to transmit one session. The file format allows storage of alternative FLUTE transmission session instructions that may lead to equivalent end results. Such alternatives may be intended for different channel conditions because of higher FEC protection or even by using different error correction schemes.

VI. System Architecture

A file delivery system using the FLUTE server file format is presented in Fig. 2. A content provider creates a media container file using a FLUTE file generator. The FLUTE server gets a copy of the media container file and uses it for compiling adata packet stream to be sent to FLUTE receivers. Needed transmission session parameters required to join, receive data from, and end FLUTE sessions can be described for example using a Session Description Protocol (SDP) [14] formatted file as is instructed in SDP Descriptors for FLUTE [15]. This file should be fetched by the FLUTE receivers before the FLUTE session begins by means of some transport protocol, such as Hypertext Transfer Protocol (HTTP) [16].

If a FLUTE receiver is not able to reconstruct the source file completely, it may contact a file repair server, typically after the the multicast/broadcast file transfer session has been completed. An example of a communication protocol and mechanism between the FLUTE receiver and the file repair server is defined in [13]. The existing source and FEC symbols included in the media container file can be used also in a post-session repair procedure between the FLUTE receiver and the repair server.

A. FLUTE File Generator Operation

A flow diagram of a FLUTE file generator operation is shown in Fig. 3. The operation starts with step $S1$ where at least one source file is provided to be added into a media

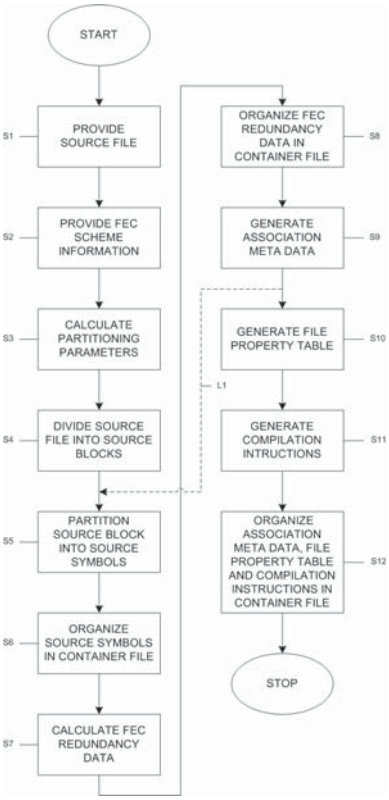

Fig. 3. Flow diagram showing a FLUTE file generator operation

container file. Information about the intended FEC scheme is provided in step $S2$, to be able to calculate partitioning parameters for the source file in step $S3$. In step $S4$, the source file is divided into source blocks according to the calculated partitioning parameters.

Each created source block is then handled with steps $S5$ through $S9$. The processed source block is partitioned into source symbols in step $S5$ also according to the partitioning parameters from step $S3$. In step $S6$, source symbols of the processed source block are organized in the media container file as a binary item according to the ISO base media file format, and the item containing the source symbols for a source block is referred to as a file reservoir. In next step $S7$, FEC symbols are calculated based on the source symbols composed in step $S5$. Next, in step $S8$, FEC symbols are organized into the media container file as a FEC reservoir item.

Next, association meta data between source and FEC symbols is generated in step $S9$. If the source file is divided to more than one source block, then steps from $S5$ to $S9$ are repeated for each source block, which is shown by line $L1$ in the figure. So, if there are N source blocks for a particular

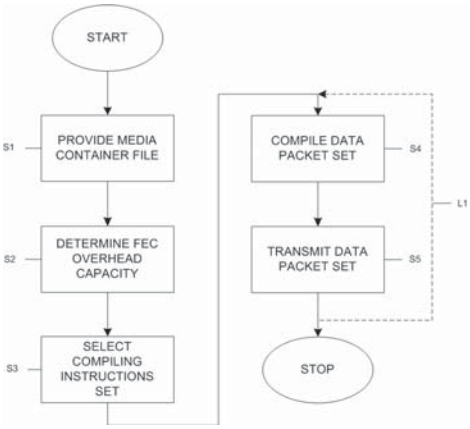

Fig. 4. Flow diagram showing a FLUTE sender operation

source file, then there exist N file and N FEC reservoir items for the particular source file in the media container file.

In next step $S10$, a file property table containing information about inserted source files is generated. In this table for example information about content type, content encoding, content length, content location, MD5 digest and file partitioning parameters for each source file can be presented. Also information about actual storage location of each reservoir items and associations between file and FEC reservoirs per each source file is usually recorded into the file property table.

Next step $S11$ generates the FLUTE packet compilation instructions to be used by media servers. These instructions can be used as hints how to compose a transmittable packet stream using the reservoir items stored in the media container file. In practice, the compilation instructions are FD hint tracks. An FD hint sample to form a transmittable packet can refer to an indicated byte range in a particular file reservoir item. Last step $S12$, organizes generated association meta data, file property table and compilation instructions into the media container file.

B. FLUTE Sender Operation

Fig. 4 shows a flow diagram of a FLUTE sender operation. The operation starts with step $S1$ where a media container file is provided to the FLUTE sender. In next step $S2$, the FLUTE sender determines FEC overhead capacity which is possible to be used with the media session in question. Using this information suitable compiling instructions set among available alternatives is selected in step $S3$. The media container file contains pre-composed source symbols and pre-calculated FEC symbols as file reservoirs and FEC reservoirs, respectively, so there is no need to source symbol construction and FEC encoding on-the-fly. Instead, the media server uses compiling instructions, meta data, and reservoir items to compile data packet set in step $S4$. Compiled data packet set is then transmitted to the FLUTE receivers in step

$S5$. It is not necessary for the FLUTE sender to compile all data packets belonging to the selected compiling instruction set at once; it is possible to repeat steps $S4$ and $S5$ several times. This is illustrated by line $L1$ in the figure.

VII. Performance Measurements

Results from performance measurements carried out by using a modified MAD-FLUTE sender [17] (based on version 1.7) with a PC (Intel Core2Duo 6300, 2GB DDR2, running Ubuntu 8.10) are presented in this section. The used media container files were generated by a FLUTE file generator implementation which comprises parts of the MAD-FCL library, a proprietary Raptor library and an enhanced proprietary MP4 file format library. Measurements have been performed using three different FEC schemes: Compact No-Code FEC [18], Reed-Solomon FEC over GF(2^8), and Raptor FEC. The impact of a systematic on-the-fly source block partitioning is studied using Compact No-Code FEC. Reed-Solomon FEC measurements demonstrate the FEC encoding effect when the source block size is relatively small. Raptor FEC measurements show how large source block sizes affect on-the-fly FEC encoding. Encoding symbol lengths and/or FLUTE packet sizes for Compact No-Code FEC and Reed-Solomon FEC measurements were set according to the Raptor FEC ones.

Table I shows elapsed times (average values from ten measurements) when the FLUTE sender was sending 100 kB, 1 MB, 10 MB and 100 MB source files, with and without a media container file, targeting to a 1 Mbps sending rate. From the table we can see that the sending time stays essentially the same regardless of the usage of the media container file with the Compact-No Code FEC and the Reed-Solomon FEC. There are a couple of reasons for this. Neither source block partitioning nor Reed-Solomon FEC encoding takes much time because the maximum source block size is relatively small, i.e., 255 encoding symbols in these measurements. The FLUTE sender is also targeting to a specific sending rate and it can catch up the small source block partitioning and/or Reed-Solomon FEC encoding times with a small decrease of the packet sending interval. The results from the Raptor FEC measurements prove the importance of avoiding on-the-fly FEC encoding with large source block sizes. Already with a 100 kB source file, when there is one source block with 1024 source symbols, the on-the-fly FEC encoding and sending takes roughly two seconds more compared to the media container file usage. The recommended minimum source block size with the Raptor FEC is 1024 source symbols. Thus, with a 100 kB source file, there are small 100 B symbols, which are then grouped into a ten-symbol FLUTE packet for transmission. The same amount of source symbols exists also with a 1 MB source file, but in this case each symbol is 1024 B long and there are more FLUTE packets for transmission. It should be noted that the used Raptor FEC library might be optimized to perform the FEC encoding faster. However, after a closer look into the library it is possible to conclude that the FEC encoding time depends heavily on the number of source symbols. On the other hand, with large source block sizes the file delivery system is much more resistant against burst errors and ought to require less overhead for similar performance.

VIII. Conclusion

As the amount of traffic in the IP based networks seems to be ever-growing, it is very important to optimize both the overhead caused by the guarantee of reliability and the load of the server in a file delivery system. As is shown in earlier studies, the use of FEC is a good option to improve the reliability from the viewpoints of receivers and the file delivery system. However, the load of the file delivery server might increase if the FEC encoding is done on-the-fly. This paper presented a file delivery system which responds to both above mentioned issues by using the FLUTE server file format.

The results from the performance measurements showed that with small source block sizes the file delivery server is able to perform source block partitioning and FEC encoding on-the-fly without an increase in the sending time. However, if a large amount of FLUTE receivers are not able to reconstruct the source file completely after the multicast/broadcast file transfer session has been completed, the file repair server might be the bottleneck if media container files are not used. Another observation from the measurements is the importance of a media container file usage with large source block sizes.

References

[1] ISO/IEC, "Information technology – Coding of audio-visual objects – Part 12: ISO base media file format," ISO/IEC 14496-12:2008, Third Edition, 2008.

[2] ——, "Information technology – Coding of audio-visual objects – Part 14: MP4 file format," ISO/IEC 14496-14:2003, First Edition, 2003.

[3] ——, "Information technology – Coding of audio-visual objects – Part 15: Advanced Video Coding (AVC) file format," ISO/IEC 14496-15:2004, First Edition, 2004.

[4] 3GPP, "Transparent end-to-end packet switched streaming service (PSS); 3GPP file format (3GP)," 3rd Generation Partnership Project (3GPP), TS 26.244, Dec. 2008. [Online]. Available: http://www.3gpp.org/ftp/Specs/html-info/26244.htm

[5] ISO/IEC, "Information technology – Coding of audio-visual objects – Part12: ISO base media file format, AMENDMENT 1: General improvements including hint tracks, metadata support, and sample groups," ISO/IEC 14496-12:2008/Amd.1, Final Proposed Draft Amendment, MPEG document N10249, 2008.

[6] M. Luby, L. Vicisano, J. Gemmell, L. Rizzo, M. Handley, and J. Crowcroft, "The Use of Forward Error Correction (FEC) in Reliable Multicast," Internet Engineering Task Force, RFC 3453, Dec. 2002. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3453.txt

[7] J. Peltotalo, S. Peltotalo, J. Harju, and R. Walsh, "Performance analysis of a file delivery system based on the FLUTE protocol," *International Journal of Communication Systems*, vol. 20, no. 6, pp. 633–659, 2007.

[8] T. Paila, M. Luby, R. Lehtonen, V. Roca, and R. Walsh, "FLUTE - File Delivery over Unidirectional Transport," Internet Engineering Task Force, RFC 3926, Oct. 2004. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3926.txt

[9] M. Luby, J. Gemmell, L. Vicisano, L. Rizzo, and J. Crowcroft, "Asynchronous Layered Coding (ALC) Protocol Instantiation," Internet Engineering Task Force, RFC 3450, Dec. 2002. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3450.txt

[10] M. Luby, J. Gemmell, L. Vicisano, L. Rizzo, M. Handley, and J. Crowcroft, "Layered Coding Transport (LCT) Building Block," Internet Engineering Task Force, RFC 3451, Dec. 2002. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3451.txt

TABLE I

ELAPSED TIMES WHEN SENDING SOURCE FILES WITH DIFFERENT FEC OVERHEADS. ISO FILE: PRE-COMPUTED SOURCE AND FEC SYMBOLS IN THE ISO FILE. NORMAL FLUTE: ON-THE-FLY SOURCE BLOCK PARTITIONING AND FEC ENCODING.

File size	Compact-No Code FEC		FEC 5%				FEC 10%			
			Reed-Solomon		Raptor		Reed-Solomon		Raptor	
	ISO File	Normal FLUTE	ISO File	Normal FLUTE	ISO File	Normal FLUTE	ISO File	Normal FLUTE	ISO File	Normal FLUTE
100 kB	104ms	105ms	109ms	109ms	109ms	2,183s	115ms	115ms	115ms	2,186s
1 MB	1,07s	1,06s	1,13s	1,12s	1,13s	2,46s	1,18s	1,17s	1,18s	2,46s
10 MB	10,73s	10,62s	11,23s	11,21s	11,27s	828,20s	11,77s	11,74s	11,80s	827,56s
100 MB	107,31s	106,26s	112,42s	112,11s	112,66s	- [1]	117,84s	117,39s	118,03s	- [1]

File size	FEC 25%				FEC 50%				FEC 100%			
	Reed-Solomon		Raptor		Reed-Solomon		Raptor		Reed-Solomon		Raptor	
	ISO File	Normal FLUTE	ISO File	Normal FLUTE	ISO File	Normal FLUTE	ISO File	Normal FLUTE	ISO File	Normal FLUTE	ISO File	Normal FLUTE
100 kB	130ms	130ms	130ms	2,19s	157ms	157ms	157ms	2,19s	210ms	210ms	209ms	2,19s
1 MB	1,34s	1,34s	1,34s	2,47s	1,61s	1,61s	1,61s	2,48s	2,15s	2,15s	2,15s	2,50s
10 MB	13,40s	13,42s	13,41s	828,03s	16,10s	16,10s	16,10s	827,78s	21,46s	21,49s	21,46s	831,63s
100 MB	134,13s	134,17s	134,13s	- [1]	160,90s	161,10s	160,95s	- [1]	214,61s	214,90s	214,61s	- [1]

[1] Not applicable with the current implementation. FEC encoding takes too much time.

[11] J. Lacan, V. Roca, J. Peltotalo, and S. Peltotalo, "Reed-Solomon Forward Error Correction (FEC) Schemes," Internet Engineering Task Force, Internet-Draft draft-ietf-rmt-bb-fec-rs-05, Nov. 2007, work in progress. [Online]. Available: http://www.ietf.org/internet-drafts/draft-ietf-rmt-bb-fec-rs-05.txt

[12] M. Luby, A. Shokrollahi, M. Watson, and T. Stockhammer, "Raptor Forward Error Correction Scheme for Object Delivery," Internet Engineering Task Force, RFC 5053, Oct. 2007. [Online]. Available: http://www.rfc-editor.org/rfc/rfc5053.txt

[13] 3GPP, "Multimedia Broadcast/Multicast Service (MBMS); Protocols and codecs," 3rd Generation Partnership Project (3GPP), TS 26.346, Dec. 2008. [Online]. Available: http://www.3gpp.org/ftp/Specs/html-info/26346.htm

[14] M. Handley and V. Jacobson, "SDP: Session Description Protocol," Internet Engineering Task Force, RFC 2327, Apr. 1998. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2327.txt

[15] H. Mehta, "SDP Descriptors for FLUTE," Internet Engineering Task Force, Internet-Draft draft-mehta-rmt-flute-sdp-05, Jan. 2006, work in progress. [Online]. Available: http://www.ietf.org/internet-drafts/draft-mehta-rmt-flute-sdp-05.txt

[16] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1," Internet Engineering Task Force, RFC 2616, Jun. 1999. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2616.txt

[17] MAD-FCL. MAD Project's Home Page. [Online]. Available: http://mad.cs.tut.fi/

[18] M. Luby and L. Vicisano, "Compact Forward Error Correction (FEC) Schemes," Internet Engineering Task Force, RFC 3695, Feb. 2004. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3695.txt

PUBLICATION P6

Jani Peltotalo, Jarmo Harju, Lassi Väätämöinen, Igor D. D. Curcio, and Imed Bouazizi, "RTSP-based Mobile Peer-to-Peer Streaming System," in *International Journal of Digital Multimedia Broadcasting*, Volume 2010, Article ID 470813, 15 pages, 2010. doi:10.1155/2010/470813

Hindawi Publishing Corporation
International Journal of Digital Multimedia Broadcasting
Volume 2010, Article ID 470813, 15 pages
doi:10.1155/2010/470813

Research Article

RTSP-based Mobile Peer-to-Peer Streaming System

Jani Peltotalo,[1] Jarmo Harju,[1] Lassi Väätämöinen,[1] Imed Bouazizi,[2] and Igor D. D. Curcio[2]

[1] *Department of Communications Engineering, Tampere University of Technology, P.O. Box 553, 33101 Tampere, Finland*
[2] *Nokia Research Center, P.O. Box 1000, 33721 Tampere, Finland*

Correspondence should be addressed to Jani Peltotalo, jani.peltotalo@tut.fi

Received 1 June 2009; Revised 12 November 2009; Accepted 6 January 2010

Academic Editor: John Buford

Peer-to-peer is emerging as a potentially disruptive technology for content distribution in the mobile Internet. In addition to the already well-known peer-to-peer file sharing, real-time peer-to-peer streaming is gaining popularity. This paper presents an effective real-time peer-to-peer streaming system for the mobile environment. The basis for the system is a scalable overlay network which groups peer into clusters according to their proximity using RTT values between peers as a criteria for the cluster selection. The actual media delivery in the system is implemented using the partial RTP stream concept: the original RTP sessions related to a media delivery are split into a number of so-called partial streams according to a predefined set of parameters in such a way that it allows low-complexity reassembly of the original media session in real-time at the receiving end. Partial streams also help in utilizing the upload capacity with finer granularity than just per one original stream. This is beneficial in mobile environments where bandwidth can be scarce.

1. Introduction

Peer-to-Peer (P2P) streaming applications are gaining more and more users around the world. These applications allow end-users to broadcast content throughout the Internet in real-time without the need for any special infrastructure, since the user's device, together with all other peers, collectively forms the infrastructure. Furthermore, dedicated servers are no longer required since every peer can serve data to other peers. This is in contrast to a service like YouTube [1] which still requires content to be uploaded to a central server first. Some of the currently existing P2P streaming applications, such as Octoshape [2] and SopCast [3], are suitable to be used in a mobile environment but still there are many issues to be solved before an optimized solution for mobile devices can be realized [4].

With real-time P2P streaming there is no need to download the entire media file before playback can be started. Decoding can be started as soon as enough data is buffered in the peer. This avoids long startup times, and eliminates the need to store the entire content on the mobile device which still has a relatively small amount of internal memory compared to the increasing size of the actual media. In live streaming, video of an ongoing event, like a football match, is delivered as a stream in real-time. After an initial buffering period, the user starts to watch the stream from a certain location and all peers consume data in the same time window. With a Video-on-Demand (VoD) streaming the user searches a video from some catalogue, and after a certain amount of initial buffering the user starts to play the video from the beginning.

In order to increase the robustness and to accommodate the limited up- and download bandwidth between peers in the network, the original multimedia session needs to be split into smaller parts, which can be reassembled at the receiving peers into the original media representation. This paper presents an effective real-time P2P streaming system where original Real-time Transport Protocol (RTP) [5] sessions related to a media delivery are split into a number of so-called partial streams according to a predefined set of parameters. This approach allows low-complexity reassembly of the original media session in real-time at the receiving end.

The structure of the remainder of this paper is as follows. The related work is discussed in Section 2. Then, a short

overview of the system is given in Section 3. Detailed descriptions of the overlay network and the media delivery are given in Sections 4–8. After that, results from the performance experiments are presented in Section 9. Interesting areas for further work are discussed and highlighted in Section 10. Finally, Section 11 concludes this paper.

2. Related Work

Many P2P file sharing applications make use of multiple source distribution. A file is first partitioned into pieces or chunks, typically of equal size. A peer then connects to the seeder or leecher peers, and requests the missing pieces of the file in a random order. The difference between a seeder and a leecher peer is that the former has a complete copy of the file while the latter has only a partial copy. For example, with BitTorrent [6] the complete multimedia file can be partitioned into blocks of 256 KB which are then selected by the interested peers and requested according to a rarest-first piece selection algorithm. This approach is not at all suitable for streaming applications as it does not consider the delay problem. Users may experience long download delays of possibly several days. It also assumes that the full content is known and available at the source peers, which does not necessarily apply to streaming applications as the stream may be live.

A P2P multimedia streaming solution based on the BitTorrent protocol is proposed in [7]. The rarest-first chunk downloading policy is replaced by a policy where peers first download chunks that will be consumed in the near future. The tit-for-tat peer selection policy is also modified to allow free tries to a larger number of peers to let peers participate sooner in the multimedia distribution. Another P2P streaming system based on a P2P file sharing implementation was proposed already in [8]. However, the data partitioning based on fixed byte ranges is not suitable for streaming a continuous media, which is of variable bit rate nature.

In P2P content distribution, an overlay network is created at the application layer in order to transfer the actual content among peers in the network. A random mesh-based overlay architecture, like in [9, 10], provides flexibility for handling peer departures, but good general connectivity between peers is not usually achieved. There have been many studies about how to organize peers in an efficient and scalable way. In [11] receivers are organized into a hierarchy of bounded-size clusters and the multicast tree is built based on that. In [12] peers are organized into a directed acyclic graph to enable peers to obtain locality awareness in a distributed fashion. To improve the file sharing performance of the BitTorrent protocol, an overlay network where peers are grouped into clusters according to their proximity is proposed in [13]. Even though some solutions have proven their functionality with wired connections, those might not be suitable for the mobile environment.

Preliminary results of the mobile P2P system described in this paper have been published in [14]. In the following sections more information about the system is given by explaining in detail the extended Real Time Streaming Protocol (RTSP) [15] messages used for signalling and providing more results from the experiments.

3. General System Overview

The architecture of the system is designed to be scalable and efficient for real-time streaming services in the mobile environment. The system supports both live and VoD streaming services. Location awareness in terms of peer proximity has been exploited to reduce delay, and thus, to improve the scalability of the system.

Peers are grouped into clusters according to their proximity in order to efficiently exchange data between peers. For VoD streaming services, the clusters could be constructed for example, based on the interest level for certain pieces of data, so that the peers watching the same part of a video at the same time belong to the same cluster. In live streaming services a cluster can be formed only based on the proximity of peers, because all peers are interested in the same data pieces within the same time window. Clusters will also help with scalability issues of peer maintenance. Peers inside a cluster are considered to be close to each other and thus communication between peers can be done more efficiently.

All overlay network operations in the system are implemented using extended RTSP messages. All RTSP methods are extended to include an additional `RTP2P-v1` tag in the `Require` header field. This tag makes it possible for the receiving peer to detect that support for the real-time P2P extensions is needed. Additionally, all RTSP messages will include a `Peer-Id` header field to indicate the source of a message. The most important new header field is called `overlay` and it is used widely in the overlay network operations. The usage of the `Overlay` header field and other additional header fields depending on the message type are explained in Sections 4 and 7. The syntaxes of the `Peer-Id` and `Overlay` header fields in Augmented Backus-Naur Form (ABNF) [16] are given below:

```
Peer-Id = "Peer-Id:" SP id CRLF id = 1*DIGIT

Overlay = "Overlay:" SP operation CRLF
operation = "backup" | "create" | "join_bcl" |
            "join_neighbor" |"join_peer" | "leave" |
            "new_peer_id" | "remove" | "split" |
            "update"
```

The RTSP Uniform Resource Locator (URL) is formatted according to the ABNF syntax shown below. The `host` and `port` parts are defined in [17, Section 3.2]. The `service-id` specifies the service and the `stream-id` specifies the RTP session. Like in [15], it is also possible to use the asterisk character instead of the URL meaning that the request does not apply to any particular resource:

```
rtsp_URL = "rtsp://" host [":" port] ["/" [ service-id
           ["/" stream-id]]]
service-id = 1*DIGIT
stream-id = 1*DIGIT
```

Peers exchange actual media data between each other using RTP. The system is using time-based chunking, which

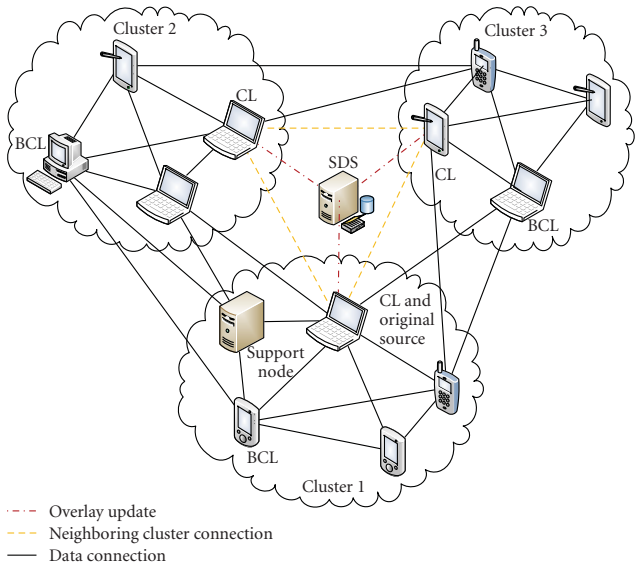

Figure 1: Overlay architecture.

creates multiple partial streams from an original RTP session. This implements multisource streaming in a way that each sender sends bursts of data from a different partial stream. Multisourcing will help to cope with the dynamics of mobile peers and distributes bandwidth usage in the system more flexibly and evenly. The original data stream source generates also RTP timestamps and sequence numbers for the RTP delivery. Timestamps and sequence numbers are delivered unchanged within the streaming service. This is done for allowing RTP packets from multiple partial streams being reassembled in the correct sequence order at each peer for local playback. The RTP time line is known system-wide, so the timestamps can be used to uniquely identify individual packets within a streaming service.

4. Overlay Network Architecture

The architecture of the overlay network with three clusters sharing a certain streaming service, such as a live stream channel or a VoD movie, is presented in Figure 1. It should be noted that for every different streaming service such an overlay network is maintained separately. The Service Discovery Server (SDS) is a central nonmobile server containing information about cluster hierarchy and the available streaming services in the system.

When a peer wants to join the P2P overlay network, a peer identifier (ID) is first requested from the SDS using an `RTSP OPTIONS` message with a `new_peer_id` tag in the `Overlay` header field. Because the peer does not have a peer ID yet, it must set the value to minus one in the `OPTIONS` message. A unique peer ID is then returned by the SDS using a `200 OK` message with a `New-Peer-Id` header field. The syntax of the `New-Peer-Id` header field in ABNF is given below:

```
New-Peer-Id = "New-Peer-Id:" SP id CRLF
id = 1*DIGIT
```

The cluster concept is implemented with the help of Cluster Leaders (CLs). There is one CL assigned to each cluster with the possibility for one or more Backup Cluster Leaders (BCLs). CLs are used to manage peers inside the cluster and to connect new arriving peers. Each ordinary peer must perform periodical keep alive messaging to inform its existence to the CL and all other peers from which it has received RTP packets. The latter is done to avoid unnecessary data transmission because RTP uses User Datagram Protocol (UDP) and the sending peer does not otherwise know that the receiving peer is still in the network. A new arriving peer can select a suitable cluster according to its best knowledge of locality using Round Trip Time (RTT) values between CLs and itself.

In addition to RTT measurements, location awareness could be also based on, for example, IP level hop count, geographic location or some combination of these three mentioned metrics. IP level hop count is not alone suitable for proximity metric, since with Virtual Private Networks (VPNs) or other tunneling techniques one hop might actually consist of a large number of hops and the distance could be quite long. Nor does small IP level hop count guarantee small delay, because it does not take connection speed into account. Geographic location is also little problematic in IP level point of view. Even if peers are geographically close to each other, the IP level routing path could circulate through distant router. Hence, only RTT values are used in our system for proximity checks.

CLs are nodes with suitable capabilities, such as a high throughput access network connection, enough memory and CPU power, and long-expected battery lifetime. One cluster should contain only a limited number of peers in order to sustain system scalability. The CL collects statistical data of the peers participating in a cluster. This statistical data contains information about service join time, reception buffer position, missing RTP packets, and upstream and downstream connections, and can be used to make the decision of the best peer from which to start downloading data. Statistical information can be used to, for example, filter out candidate source peers which already have many upstream connections or lots of missing RTP packets. Service join time can be used to estimate the behavior of the peer. If the peer has joined to the service very long time ago, it is most likely a stable peer which will provide data in the future also. On the other hand, without extra information about the expected battery lifetime with mobile devices, long service joining time can also mean short-expected battery lifetime.

The CL is a functional entity in the network and may also participate as an ordinary peer at the same time, by receiving and sending media data. Thus, the CL can be seen as a functional extension of an ordinary peer. The CL will inform the SDS currently at ten seconds intervals about changes in the cluster by sending an `OPTIONS` message with an `update` tag in the `Overlay` header field in order to maintain an up-to-date cluster list at the SDS. The updated cluster information will be expressed using an Extensible Markup Language (XML) [18]. To decrease the amount of data delivered in the network, all XML fragments are compressed using `deflate` compression mechanism from the zlib data compression library [19].

While joining the selected cluster, a peer receives an initial list of peers from which the actual media data can be acquired. Naturally, the corresponding CL inserts joined peers into its peer list, and if the amount of peers is very large, the CL can return only a subset of peers. Proximity testing in the peer selection is optional since the cluster selection procedure guarantees that peers are reasonably close to each other. Anyway, a peer which finally selects its sources for the stream, needs to test a certain amount of peers until suitable ones are found. The peer can later receive updates of the peer list while performing periodical keep alive messaging to the CL, which ensures that the peer list can be kept up-to-date during the streaming service.

The peer's contact information, that is, all information needed for contacting the peer, could include also a cluster ID, so that peers can prioritize connections within their own cluster. However, there should always be data connections between peers that are located in different clusters. This ensures that clusters do not become separate islands having only one incoming connection from other clusters, which would form a single point of failure that could cause problems later on when that peer leaves the streaming service.

4.1. Service Creation and Initial Cluster. The message exchange during service creation is presented in Figure 2. When a peer wants to create a service, an `ANNOUNCE` message will be sent to the SDS. A `Client-Port` header field indicates the port number to be used in the overlay communication. The service is described using the Session Description Protocol (SDP) [20]. Two new SDP attributes, `service-type` and `stream-info` are used to signal the service information. The `service-type` attribute defines the type for the service, and the `stream-info` attribute defines the identifier for the RTP session and parameters to be used in the RTP session partitioning explained in Section 7. The syntaxes for the `service-type` and `stream-info` attributes and the `Client-Port` header field in ABNF are given below:

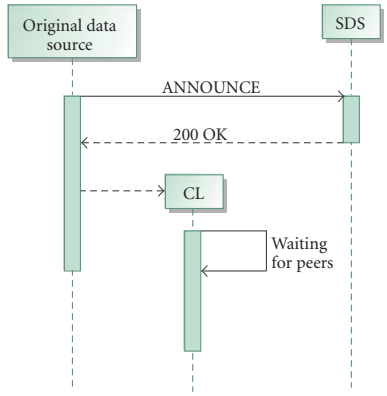

Figure 2: Creation of the service and the initial cluster.

```
service-type-line = "a=service-type:" type CRLF
type = "live" | "vod"

stream-info-line = "a=stream-info:" id";" piece-size";"
                   nb-of-partials ";" CRLF
id = "id=" 1*DIGIT
piece-size = "piece-size=" 1*DIGIT
nb-of-partials = "nb-of-partials=" 1*DIGIT

Client-Port = "Client-Port:" SP port CRLF
port = 1*DIGIT
```

As a response to the successful session creation, a `200 OK` message is sent by the SDS. The message contains the `Cluster-Id` and `Service-Id` header fields to describe the IDs for the initial cluster and the newly created service, respectively. A `301 Moved Permanently` message can also be sent if the SDS has been moved to another location. In a redirection case the `Location` header field must be present informing the new location of the SDS. Any other message type must be interpreted as a failed session creation. The syntaxes of the `Cluster-Id` and `Service-Id` header fields in ABNF are given below:

```
Cluster-Id = "Cluster-Id:" SP id CRLF
id = 1*DIGIT

Service-Id = "Service-Id:" SP id CRLF
id = 1*DIGIT
```

There are two possibilities for creating the initial cluster and selecting a CL for it: (a) the first peer joining the service

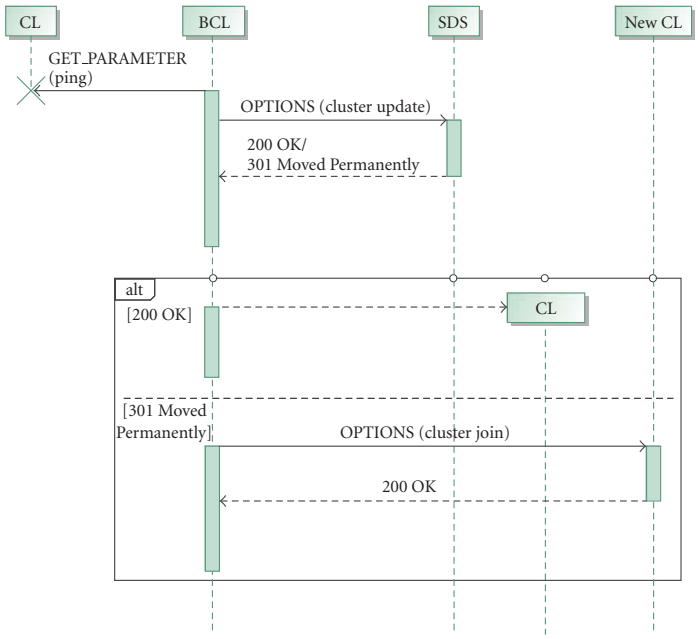

Figure 3: Uncontrolled CL departure.

will be assigned as a CL by the SDS, and (b) the original data source will be the first CL in the service. The latter possibility uses more resources from the original data source and therefore the original data source should be released from the CL responsibilities when possible. However, the latter possibility also guarantees that the initial cluster remains operational because the first joining peer might depart from the service quite quickly. Hence, the alternative (b) is used in our system.

When the service is successfully created, the original data source becomes CL for the initial cluster, which is illustrated by the dashed line without message type in Figure 2, and starts to wait for other peers to join the service. When new peers are joining the service, BCLs are assigned by the CL by using an `OPTIONS` message with a `backup` tag in the `Overlay` header field. If a peer accepts the BCL assignment, it sends a `200 OK` message, and if not, it will send a `403 Forbidden` message.

The service is updated and removed by using an `ANNOUNCE` message. If some part of the information have changed, the SDS updates the information in the database. To remove a service, the stop time in the SDP `t-line` should be set smaller than the prevailing system time, which means that the service has been stopped and the SDS can remove the service from the database. To a successful service update or removal the SDS will respond with a `200 OK` message, otherwise the SDS will return `400 Bad Request or 404 Not Found` messages.

4.2. Cluster Leader Departure. When the CL leaves the network it needs to be replaced by one of the BCLs. If a cluster does not have an active CL, new peers cannot be accepted into the network. However, this does not affect the data streaming connections between existing peers because the streaming and overlay connections are independent. New peers cannot be discovered by normal peers during the cluster leader change, but this should not be an issue because peers should have knowledge about more peers than they are using.

The message exchange in the event of an uncontrolled CL departure is presented in Figure 3. When the BCL does not get a response to its periodical `GET_PARAMETER` message, it concludes that the CL has left from the cluster and contacts the SDS using an `OPTIONS` message with an `update` tag in the `Overlay` header field to replace the old CL. The source of the first received `OPTIONS` message will be assigned as a new CL, illustrated by the dashed line without message type in the figure, and the new arriving peers can normally start using the new CL. All other BCLs will receive a `301 Moved Permanently` message with the information about the new CL and will send an `OPTIONS` message with the `join_bcl` tag in the `Overlay` header field to the new CL and will continue in the BCL role. If the original CL has not left the cluster but has had connectivity issues, it is redirected to the new CL by the SDS. In this case the old CL becomes a BCL.

When a peer notices that the CL is not available, it tries to connect to the known BCLs. If the BCL has replaced the

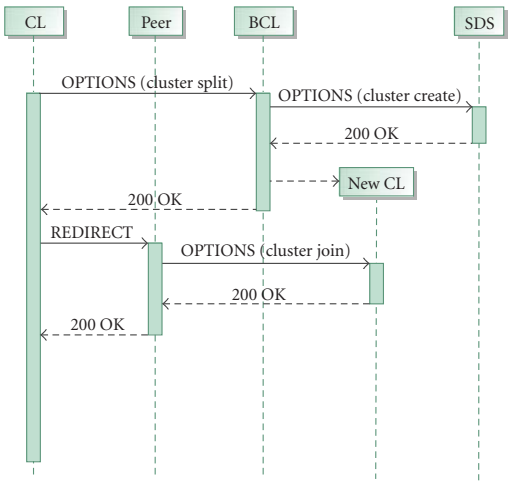

Figure 4: Successful cluster splitting procedure.

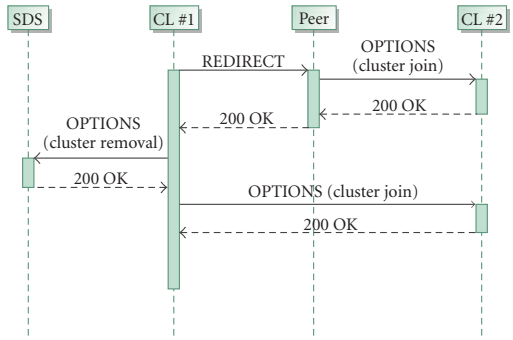

Figure 5: Successful cluster merging procedure.

old CL, it accepts the connection with a `200 OK` message, otherwise it sends a `301 Moved Permanently` message with the `Location` header field indicating the location of the last known CL. It should be noticed that the CL departure is not an atomic operation and takes some time, and therefore there can be short times when any one of the BCLs does not know the correct CL of the cluster. If none of the BCLs in the list respond, the peer sends a query to the SDS and asks for a new cluster which it can join.

4.3. Cluster Splitting and Merging. When the cluster grows too large to be handled by a single CL, the cluster should be split into two separate clusters. The existing CL assigns one of its BCLs to become a new CL for the new cluster, and redirects a number of existing peers to the new cluster. The message exchange in the successful cluster splitting procedure is presented in Figure 4.

Cluster splitting is performed by using an `OPTIONS` message with the `split` tag in the `Overlay` header field. After receiving this message, the BCL will inform the SDS that wants to become the CL of a new cluster by sending an `OPTIONS` message with a `create` tag in the `Overlay` header field. To a successful cluster creation, the SDS will respond with a `200 OK` message, which contains a `Cluster-Id` header field to describe the ID for the new cluster. Otherwise, the SDS will return a `400 Bad Request` message if the request message format is not valid, or a `404 Not Found` message if the service is not available anymore. After a successful cluster creation, the BCL will become the CL of the new cluster, and replies to the splitting CL by sending a `200 OK` message, otherwise it must send a `400 Bad Request` message to the splitting CL and wait for further instructions.

The splitting CL then sends a `REDIRECT` message, with the location of the new CL in the `Location` header field to those peers that should change the cluster. The redirected peers will then join the new cluster by sending an `OPTIONS` message to the new CL. After a successful cluster join, that is, the peer received a `200 OK` message from the new CL, the peer will send a `200 OK` message to the splitting CL. Otherwise, the peer will send a `400 Bad Request` message to inform that it is not possible to join to the new cluster.

The overlay connections between the CLs are created after a successful splitting by sending an `OPTIONS` message with a `join_neighbor` tag in the `Overlay` header field and a `200 OK` message. This connection is subsequently used to exchange cluster information expressed using XML fragments between neighboring clusters.

Merging of two clusters must be done when a cluster becomes too small. If the amount of peers is too small, a new joining peer will get a very small list of data sources which makes the functionality less reliable when one of these peers leaves the service. The message exchange in the successful cluster merging procedure is presented in Figure 5.

The merging is started by the CL, when the amount of peers in the cluster drops below some predefined threshold, by sending a `REDIRECT` message to all peers in the cluster. Peers will then join the new cluster, selected by the merging CL from its neighbor clusters, by sending an `OPTIONS` message to the new CL. After a successful cluster join, the peer will send a `200 OK` message to the merging CL. If the redirected peer does not receive any response from the new CL or it receives a `400 Bad Request` message, it must send a `400 Bad Request` message to the merging CL to inform that it is not possible to join to the new cluster and wait for further instructions.

After all peers in the cluster have confirmed the cluster change, the merging CL will remove the cluster by sending an `OPTIONS` message with a `remove` tag in the `Overlay` header field to the SDS. This message is optional, because the cluster is removed also automatically if a keepalive message has not been received during a certain interval. To a successful cluster removal, the SDS will respond with a `200 OK` message. Otherwise, the SDS will return a `400 Bad Request` message if the request message format is not valid, or a `404 Not Found` message if the cluster is not available anymore. The merging CL itself then must send a cluster join message, that is, an `OPTIONS` message with a `join_bcl` tag in

the `Overlay` header field, to a known neighbor and join as a BCL.

All overlay network connections are maintained by sending `GET_PARAMETER` and `200 OK` messages between peers as keep alive messages. Keep alive messages between neighboring CLs are exchanged at 20 seconds intervals and are used to exchange information about neighboring clusters. Keep alive messages between the CL and the BCL are used to deliver cluster information to the BCL and these messages are sent at 30 seconds intervals.

5. Partial RTP Streams

In order to have unique sending slots for each of the sources, a partial RTP stream concept is introduced in this paper. First, every RTP session, such as video, audio or subtitle streams of the entire multimedia session is split into smaller pieces along the time axis. Each of the pieces has a fixed duration T_P which is expressed in time. The start time is aligned with the start time of the RTP time base, that is, the start of the first piece is located at the origin of the RTP time line. One of the benefits of taking time as a unit is that all packets can remain intact at the RTP layer. Segmentation at the RTP packet level is not required for creating the partial streams. This significantly reduces the complexity of the implementation.

In the second step, RTP packets belonging to each of the RTP sessions are assigned to N partial streams according to (1), where i denotes the index of the partial stream ($0 \leq i < N$) and t_{RTP} denotes the RTP timestamp as carried in the RTP data packet. The algorithm allows assigning every RTP packet in the session to a partial RTP stream without having to maintain the state in the peer itself. Only by examining the RTP timestamp in combination with the constant parameters is sufficient to identify the partial stream the RTP packet is assigned to. Assuming a timeline of a single RTP session, this process is illustrated in Figure 6, where $T_C = N * T_P$ is used to denote the cycle time:

$$i = \text{floor}\left(\frac{t_{\text{RTP}}}{T_P}\right) \bmod N. \tag{1}$$

Note that the piece size T_P should be selected in such a way that it is large enough to contain at least one RTP packet on average. If it is chosen too small, not every piece will have data, which may in the extreme case lead to an empty partial stream. On the other hand, larger cycle times lead to longer startup times, since a complete cycle needs to be buffered before seamless playback can be guaranteed. In one particular type of partitioning, every piece would start with an intracoded picture. This would facilitate independent decoding of partial streams in the presence of packet loss due to the fact that a partial stream is not being received. This could for instance easily be achieved by aligning the pieces with group-of-picture (GOP) boundaries. Enhanced robustness will also be achieved by assigning key (RTP) packets to multiple partials. Key packets could for instance be Instantaneous Decoding Refresh (IDR) picture data or other data that would help error concealment. Duplicate RTP packets would simply be removed upon reception and would therefore not affect the basic algorithm or its complexity.

The number of partial streams, N, can vary per RTP session. For instance it may not be very useful to partition an audio stream into lower bit rate partial streams if the bit rate of the entire RTP audio session is already in the order of magnitude of a single partial RTP video stream. This also has the additional advantage that the audio is received either in its entirety or not at all, thereby reducing annoying audible artifacts in the case of partial stream loss.

The number of partial streams does not necessarily need to be constant throughout the P2P network within a particular streaming service. As a matter of fact it is possible to vary N at every forwarding peer in the network. However, choosing the same N throughout the network simplifies the design of the partitioning functionality.

6. Media Delivery Mechanism

All peers in the streaming network are forming a non-hierarchical mesh structure. Peers are connected to several other peers and are receiving data from and sending data to multiple other peers. An example mesh layout can be seen in Figure 1.

A peer may request the delivery of one or more partial streams from another peer. A partial stream is the smallest granularity for media streaming, that is, a peer may not stream a fraction of a partial stream. The number of partial streams can be tuned to achieve the target bit rate of a partial stream. Each peer in the network should have enough uplink bandwidth to be able to stream at least a single partial stream.

In this kind of streaming network, where most of the peers are from the same cluster, some kind of intelligence to avoid loops is needed. Such loops occur when a sender starts receiving its own data via a number of intermediate peers in the mesh network. To avoid loops, an algorithm based on streaming path in the form of list of ancestors is used. The path for the data stream in the application level containing peer IDs which have forwarded the stream is delivered using the contributing source (CSRC) list in the RTP packets. This list is then used to avoid accepting connections from peers who are already in the list and for dropping connections if a peer notices that it is in the list.

Figure 7 illustrates media delivery among four peers. Arrows between each peer denote an active RTP session, and the direction defines the data flow direction. A sourcing peer can send multiple partial streams to a particular receiving peer. This allows for a smaller granularity of rate adaptation between two individual peers in the network. These multiple partial streams could either be streamed in a single RTP session or separate RTP sessions. Peers in the figure are numbered and colored. The smaller the number is, the earlier the peer has joined the network; in this case peer number one is the original source. Different colors in the peers buffers show the origin of the received data. For simplicity, the value four is used for the number of partials for each peer.

In order to receive a complete stream, a peer must receive all partial streams; in this example, partials 0, 1, 2, and 3.

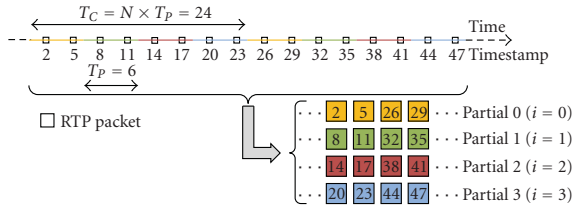

Figure 6: RTP stream partitioning.

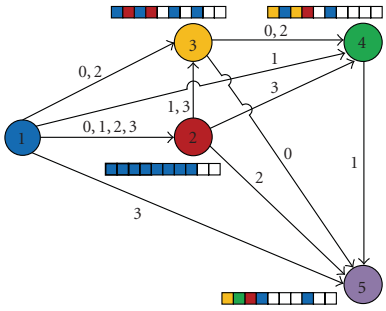

Figure 7: Partial RTP stream delivery.

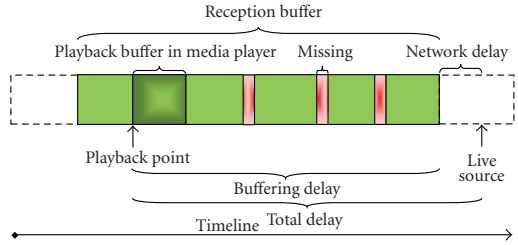

Figure 8: Data buffering.

Peer number two is receiving all partials from the original data source, that is, peer number one, and is forwarding partials to peers number three, four, and five. Peer number four is also receiving partials from peer number three, and peer number five from peers number three and four. All incoming packets for all of the partial streams that constitute a particular RTP session are added to a separate buffer pool. This buffer pool maintains the information about the destination peer to which the incoming packets need to be forwarded. Every incoming packet is examined and assigned to one of the outgoing queues. In case the peer is playing back the received streams locally, the local player is considered to be a destination as well.

Packets destined for a particular receiving peer are transferred from the buffer to the outgoing RTP queue as soon as they have been received. In case of local playback, that is, the receiving peer is decoding and rendering the received RTP session, the packets are also forwarded internally to the media player. Buffering is needed to recover from peer departures and consequential missing data. The peer can ask a replacement peer to send the missing data from its buffer. There is also a possibility that a peer does not have certain part of the buffer available because of bad network conditions, for example, insufficient bandwidth. These parts might be later downloaded if needed. Because of the missing data, the playback is interrupted during this period and it depends on the used codec how missing data will affect on Quality of Experience (QoE). Data will most probably not come in order or in time from all sources as the delay could vary very much between different sources, so the buffer is also needed to collect all the data from different partial streams and arrange the data in order before passing it to the local media player.

When all partial streams are received almost at the same time, the buffering delay can be reduced and the data can be passed earlier to the media player. However, the situation may change during time, and a constant buffer delay is currently used in the implementation. Data buffering is presented in Figure 8. The playback buffer is the buffer located in the media player. When peers consume media, the reception buffer is simultaneously shared with other peers. The total delay from content generation to receiver playback is the sum of the network delay, buffer delay, and media player playback delay.

In addition to the reception buffer, data storage via caching should be used in a VoD streaming service. When using caches, the VoD data can be distributed away from the original data source. This helps relieving the network load from the original data source, as new peers joining the network are able to download VoD content from multiple sources instead of relying on the original data source. Caching could be implemented in many ways, in the simplest model all peers store all data they have consumed. If a peer does not have enough storage capacity, the data belonging to a specific partial stream could be cached instead. Alternatively, a peer can limit the amount of cached data by applying a sliding time window. In the former case, an algorithm based on peer IDs could be used to determine which partial stream should be stored. This kind of partial caching requires that the amount of peers in a streaming service is reasonably high. To ensure data availability from multiple peers in every situation, support nodes, which

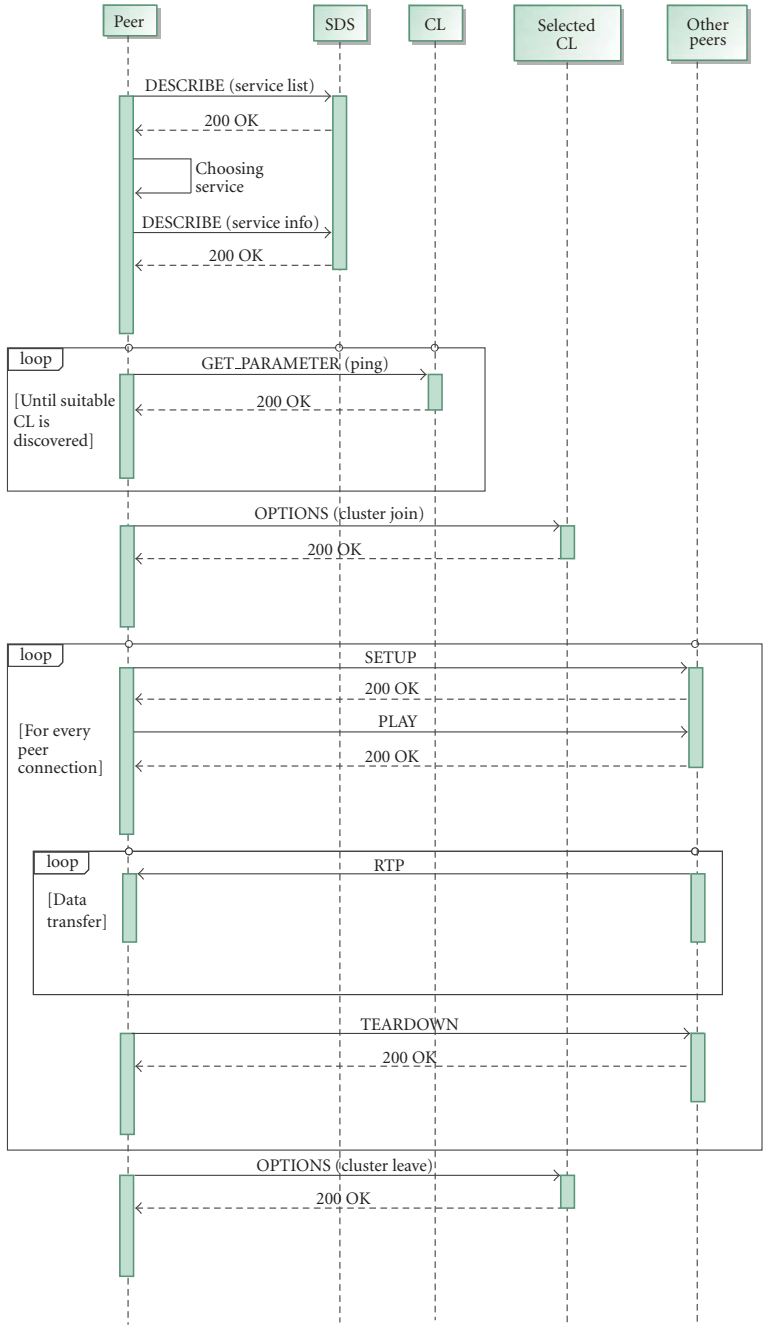

Figure 9: Peer in a service.

act as stream relay nodes, could be used to store all the data. Such relay nodes simply store the data and pass it to other network participants. It should be noted that a support node also consumes upload bandwidth from other peers, but with a high throughput network connection it provides more bandwidth to others than it consumes. The currently existing simple VoD implementation uses the simplest caching model, which is of course not the most optimal one, but it enabled a fast proof-of-concept implementation.

7. Peer Operation in a Service

Figure 9 shows the message exchange in case a peer is participating in a particular service. For getting a list of all available services a `DESCRIBE` message is sent to the SDS. The service list information obtained in the `200 OK` message contains only general information of the services to decrease the message size and the information is expressed as XML fragments. If a user wishes to search services with a wildcard string, a search element could be used to deliver the wildcard string to the SDS. If there are not available services, the SDS will respond with a `404 Not Found` message.

To be able to join to a particular service, more detailed service information must be retrieved from the SDS using a `DESCRIBE` message with an RTSP URL specifying the service. A `200 OK` message contains only a partial list of the available clusters, in case a large number of clusters has been created. The response uses multipart MIME [21], because it must deliver both the SDP of the service and the initial cluster list in XML format. If the service is not available anymore, a `204 No Content` message will be sent by the SDS.

After obtaining the CL list from the SDS, the peer makes contact with several CLs until a suitable CL is discovered. For this purpose, the peer sends a `GET_PARAMETER` message to the CL and starts the RTT counter. The peer stops the RTT counter when it receives a `200 OK` message and compares the RTT value to some predefined upper limit. The first CL with an RTT value below the limit is then selected as the CL for the service.

The peer joins the selected cluster by sending an `OPTIONS` message with a `join_peer` tag in the `Overlay` header field to the CL of the cluster. In the `200 OK` message, an initial peer list is received in XML format if some peers have already joined the cluster. Otherwise, a `Cluster-Id` header field is used to describe the ID for the cluster. The initial peer list is a random subset of the total peer set, if there are lots of peers in the cluster.

The peer tries to request data from other peers in the list order using a `SETUP` message. This message handles the configuration of the UDP port numbers for the RTP reception using the `Transport` header field. If there are fewer peers than the target number of partial streams, the peer continues requesting again from the beginning of the list, so that it will receive more than one partial stream from a single peer. If a certain peer is not responding, it will be removed from the internal *known peer list*, so that the peer does not try to reconnect again. The peer which is receiving the requested stream, that is, audio or video stream, will respond with a `200 OK` message to indicate that it might be possible to get the stream data from the peer.

Setting up of the partial RTP streams is done by sending a `PLAY` message to the peer which is receiving the requested stream. Splitting of the original RTP session into partial streams is explained in Section 5, and these `partial stream` parameters are signaled using a `Partial-Stream` header field. The format of a `Partial-Stream` header field in ABNF is given below:

```
Partial-Stream = "Partial-Stream:" SP partial-stream-
                 info
                 CRLF

partial-stream-info = id ";" piece-size ";"
                      nb-of-partials ";"
id = "id=" 1*DIGIT
piece-size = "piece-size=" 1*DIGIT
nb-of-partials = "nb-of-partials=" 1*DIGIT
```

If the peer is able to send the requested partials, it will respond with a `200 OK` message. If the peer noticed a loop, it will response with a `400 Bad Request` message, and if the peer cannot send the requested partials, it will respond with a `404 Not Found` message. After the `200 OK` message, data delivery from the requested peer using RTP is started. If the interval between two consecutive RTP packets is more than the predefined maximum allowed delay, the receiving peer should conclude that the sending peer is not capable of delivering the data in time and it should change the sender for the partial in question.

A peer can depart from the network in two ways. In a controlled departure, the peer informs its neighbors and the CL that it is leaving the network. The peer sends an `OPTIONS` message with a `leave`, tag in the `Overlay` header field to the CL, and a `TEARDOWN` message to all of its data delivery neighbors. Neighbors, which were sending data know that they can terminate the RTP session. Also peers that were receiving data know that they will not be able to receive more data from that peer, and can search for a replacement. The `TEARDOWN` message will also be sent when a peer notices that there is a loop in the data delivery for some partial stream.

An uncontrolled peer departure is noticed both by the CL and a peer which sends data to the departed peer after connection keep alive messages, that is, `GET_PARAMETER` messages, have not been received within some time interval X. Currently, keep alive messages are sent at 30 seconds intervals towards the CL and at 15 seconds intervals towards the peer which sends the data. If the sending peer does not receive keep alive message within 30 seconds interval it concludes that the receiving peer has departed and terminates the RTP session. Similarly, if the CL does not receive any message from the peer within 45 seconds interval it concludes that the receiving peer has departed and removes the peer from the cluster because of inactivity. A peer which is receiving data from the departed peer will notice uncontrolled departure after it has not received any RTP packets since Y seconds. The value Y should be defined so that it is possible to get data from a replacement peer within the reception buffer duration in order to avoid interruption. This is basically the same situation when the sending peer

is not capable of delivering the data in time and currently Y is calculated according to (2). This value consists of time between two pieces belonging to the same partial RTP stream, the normal network delay T_N, that can be calculated from RTSP request-response pairs, and a small extra time T_E given to peers to patch packets that might still be forwarded in the network:

$$Y = T_P * (N - 1) + T_N + T_E. \quad (2)$$

To request data from a replacement peer from a certain starting point, a `Packet-Range` header field can be included into a `PLAY` message to signal the play-after value using RTP sequence numbers. The `Packet-Range` header field can also be used to signal the current playback position when the peers are seeking a new playback position in a VoD service. The format of a `Packet-Range` header field in ABNF is given below. The two different use cases can be distinguished by the minus sign used in the former case:

```
Packet-Range = "Packet-Range:" SP range-specifier CRLF
range-specifier = 1*DIGIT ["-"]
```

In the VoD service, two other additional header fields are required for the seeking operation. The desired seek time in milliseconds will be signaled using a `Seek` header field. A `Fast-Send` header field will be used to inform the sender to send a specified amount of data (in milliseconds) as fast as possible to be able to fill up the reception buffer and start playback with as small delay as possible. The formats of the `Seek` and a `Fast-Send` header fields in ABNF are given below:

```
Seek = "Seek:" SP seek-time CRLF
seek-time = 1*DIGIT

Fast-Send = "Fast-Send:" SP fast-send-time CRLF
fast-send-time = 1*DIGIT
```

8. Implementation

The architecture of the real-time P2P streaming (RTP2P) application is presented in Figure 10. The RTP2P application is implemented using C++ and it consist of ten different software components. The principle in the figure is that the higher layer uses all components that are immediately below it, so, for example, the Graphical User Interface (GUI) uses Service, Common, and Media Player components.

The GUI is implemented using the gtkmm framework [22] for the Linux desktop environment and the maemomm framework [23] for the Nokia N800 Internet Tablet. Currently three different media players, VLC [24], MPlayer [25], and GStreamer [26], are needed in the application. This is necessary because any single player cannot offer all the features required for our application. VLC is used to stream RTP packets locally to the RTP2P application in the case of the original data source. The original data source only listens to the local socket, and receives RTP packets generated by the VLC and can forward those further using the multisource streaming concept explained in Sections 5 and 6. MPlayer is used for the media playback on the client applications. It is

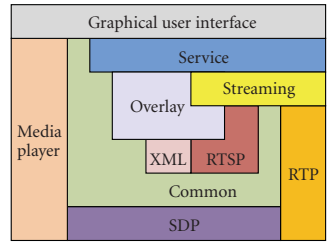

Figure 10: Architecture of the RTP2P streaming application

Figure 11: Graphical user interface.

also possible to use VLC for the media playback, but with MPlayer it is possible to achieve a better synchronization with multiple elementary streams by using RTCP sender reports. GStreamer is used to create an RTP stream from the camera of the N800 device. The needed RTP operations are provided by the GNU ccRTP library [27]. In addition, the Boost libraries [28] are utilized for threading time and file system operations.

Other proprietary software components, that is, Service, Overlay, Streaming, XML, RTSP, Common, and SDP, form the basis for the peer operation. The SDP component is based on the GNU oSIP library [29] and is used to parse the streaming service description expressed in the SDP format. The Common component contains definitions and functionalities which are widely used by other components including, for example, threading and socket operations. The XML component is based on the Expat XML parser [30] and it is utilized for parsing cluster and service information expressed in the XML format. The RTSP component contains RTSP message creation and parsing and also the base functionality for RTSP operations, which are enhanced and utilized by Streaming and Overlay components. The Streaming component includes sender and receiver functionalities to handle P2P RTSP communications and RTP reception and sending operations for the streaming service. The functionality of the cluster leader and all RTSP overlay communication are included in the Overlay component. The Service component contains the functionality needed to join, create, and manage streaming services.

Figure 11 presents the GUI in a Nokia N800 device. The GUI consist of three parts: (a) the main application view

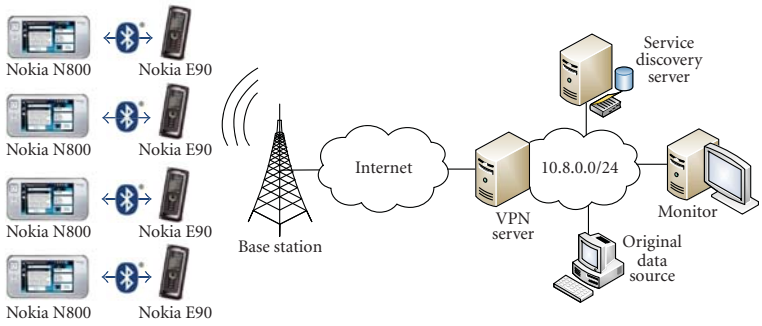

Figure 12: Test setup for evaluating the operation of the system in the mobile environment.

with Main, Info and Player tabs, (b) Create, Connect, List and Preferences dialogs launched from the drop down menu, and (c) Toolbar buttons at the bottom to start and stop the selected service at the receiving end, and the service in the original data source. The List dialog is used to obtain the service list from the SDS which is then shown in the Main tab. Information about the sourced service is also shown in the Main tab. The Info tab is used to show detailed service information about the sourced service and the service that is currently received by the peer. The Player tab shows the sourced stream in the original data source and the currently received stream by the peer. The Connect dialog is used to enter an IP address or a domain name for connecting to the SDS. The Create dialog is used for creating new services. Service name, type and description are given by the user, as well as the file to be sourced in case of stored content. The content can also be captured directly from the camera. The Preferences dialog can be used to adjust some attributes, for example, port numbers for the network traffic and the level of debug messaging.

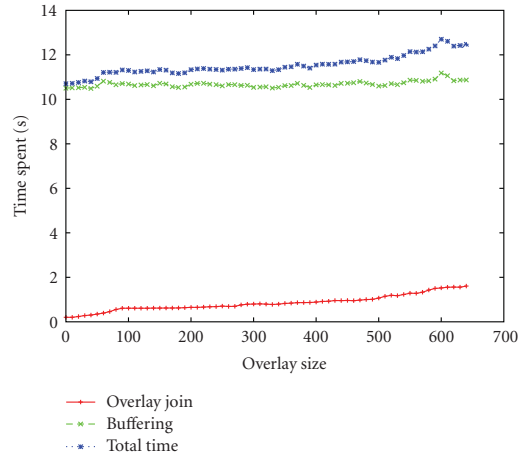

Figure 13: Overlay joining time and an initial buffering time as a function of overlay size; maximum cluster size 30 peers.

9. Performance Evaluation

The test setup for evaluating the operation of the system in the mobile environment is presented in Figure 12. Four Nokia N800 Internet Tablets with HSDPA network connection provided by the Nokia E90 were used together with a PC acting as an SDS. In some test cases, also another PC was acting as an original data source. During the tests a special application was used to monitor streaming connections between peers. As most normal consumer connections, also mobile connections are suffering from Network Address Translation (NAT) based connection limitations because Internet Service Providers (ISPs) do not want that users use the relatively small mobile bandwidth for hosting services or using P2P technologies. To avoid connection limitations, a workaround with VPN solution was utilized. Every entity with a restricted network connection first creates a connection to the VPN server and gets the desired free connectivity through the VPN tunnel. A more permanent solution could be implemented using NAT traversal to overcome these restrictions.

Tests have shown that the system performs well over mobile connections with ten seconds initial buffering time. The buffer size in our application is also ten seconds due to the RTP usage and is relatively small compared to the buffer sizes of existing solutions reported in [4].

Because the amount of available mobile devices was limited, a laboratory network environment with 17 desktop PCs (Intel Core2Duo E6550, 4 GB DDR2, running CentOS Linux, kernel 2.6.18 PAE) has been utilized to test the system functionality with a larger amount of peers. 16 hosts were used to run 40 peers in each host together with one host acting both as an SDS and as an original data source. The connectivity between all devices was provided by a 1 Gbps switch. The length for one live streaming service was roughly one hour and all forthcoming figures present average values from five different live streaming services. The maximum cluster size was set either to 30 or to 70 peers, and peers were started in 40 cycles with a 5 s starting interval: first one peer was started at each host, then a second one and so on.

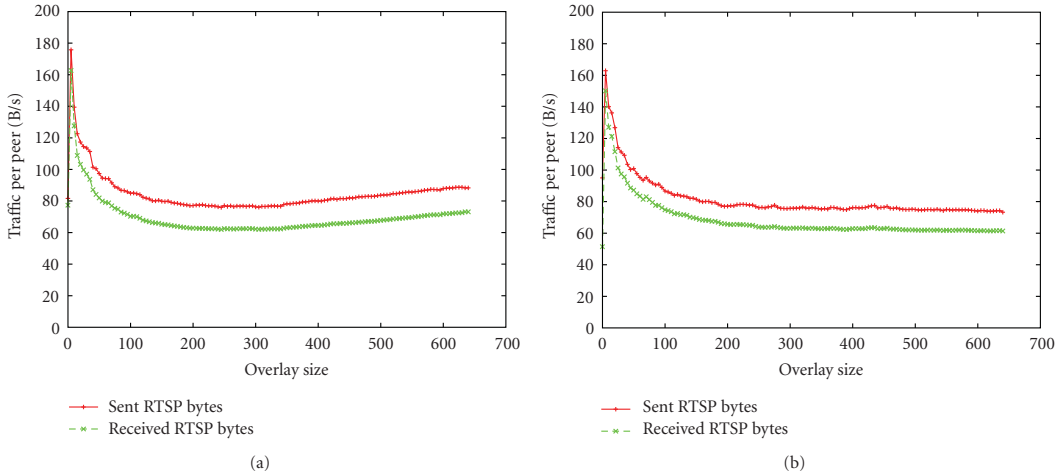

Figure 14: The amount of sent and received RTSP data, (a) maximum cluster size 30 peers, and (b) maximum cluster size 70 peers.

9.1. Steady Network. In this test scenario all peers stayed steadily in the service from the joining time to the end of the service. The streaming service joining time, that is, the overlay joining time plus the initial buffering time, as a function of the overlay size is presented in Figure 13. From the figure we can see that the initial buffering time remains almost constant regardless of the number of peers in the network. The increase in overlay joining time could be minimized by improving the current cluster selection algorithm.

The amount of sent and received RTSP data in bytes per peer as a function of overlay size is shown in Figure 14. The combined bit rate of the original RTP sessions is about 112 kbps, encoded using FFmpeg's [31] H.263+ video, and AAC audio codecs. Hence, the RTSP signalling overhead is quite minimal compared to the actual media data. The reason for the larger amount of RSTP data sent is caused by the relatively large cluster status information messages transmitted to the SDS by the CLs. The main difference between Figure 14(a) and Figure 14(b) is the slight increase in the RTSP signalling data after 300 peers with the maximum cluster size of 30 peers. This is caused by the larger (and increasing) amount of clusters and cluster status information messages. Similar effect might take place also with the maximum cluster size of 70 peers when the amount of peers is increased above the currently used maximum value.

9.2. Network with Leaving and Rejoining Peers. To simulate even a more realistic situation and the churning caused by the real mobile nodes, a timer functionality that is able to randomly shut down and restart nodes was used. After a peer had joined to the service, it stayed randomly from 30 s to 10 min in the service and then left the service and joined back after 10 s. This allowed us to test the network in more realistic situations where peers are leaving and data connections are failing.

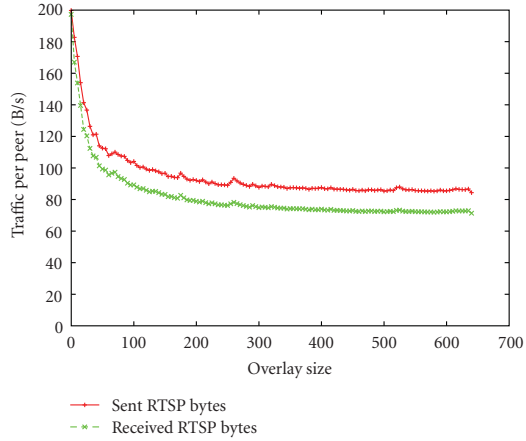

Figure 15: The amount of sent and received RTSP data with leaving and rejoining peers; maximum cluster size 70 peers

The amount of sent and received RTSP data in bytes per peer as a function of the overlay size is presented in Figure 15. If we compare these values to the corresponding values in the steady state scenario, we can see that the values follow the same trend, but the replacement peer searching and peer rejoining messages cause small overall increase to the signalling data.

10. Future Developments

In the current overlay implementation, a cluster change is controlled by the CL. Peer-controlled cluster change, where a peer changes the cluster after acquiring knowledge about a new cluster which would serve its data requirements better,

would make the system more scalable and will increase the overall performance.

Clusters are currently loosely connected together between the neighboring CLs to share some peer information. A clearer cluster group structure with a Cluster Group Leader (CGL) is worth studying. The CGL could collect the information about all clusters within the cluster group and send cluster update messages to the SDS, instead of separate update messages from individual CLs. This organization into an n-level hierarchical structure could increase network scalability and reduce the overlay joining time since the cluster search time would be reduced from $O(n)$ to $O(\log(n))$. As a drawback, the complexity of the system and the overlay maintenance will be highly increased.

More advanced implementation level support for VoD streaming such as better caching mechanism and support for other Video Cassette Recording (VCR) functionalities like fast-forward and rewind, in addition to the currently existing seek functionality will definitely pose different requirements compared to the live streaming case. Mechanisms for handling packet losses is an important research area in peer-to-peer streaming. Different error robustness techniques, such as simple retransmission, Forward Error Correction (FEC) and network coding, need to be studied to find out the benefits and drawbacks of those techniques, when used in addition to the current mechanism which is based on peer replacement before the reception buffer underflows.

One interesting research area is the usage of Multiple Description Coding (MDC) [32] or Scalable Video Coding (SVC) [33] in the real-time P2P streaming as is proposed also in [10]. A single stream is divided into several descriptions and each of the descriptions is then forwarded separately to the network. With this approach, the current partial stream could be replaced by one description without affecting the clustered overlay network architecture. However, our partial stream concept has much lower complexity (than MDC or SVC), which has enabled a fast proof-of-concept implementation. Our design allows an easy replacement of the partial stream concept with MDC or SVC as soon as their implementations become publicly available.

11. Conclusions

The effective real-time P2P streaming system for the mobile environment presented in this paper is an alternative solution to traditional client-server-based streaming applications. A scalable overlay network which groups peers into clusters according to their proximity is created and maintained using extended RTSP messages by the cluster leaders with the help of a service discovery server. Furthermore, the actual media delivery is implemented using a partial RTP stream concept. RTP sessions are split into a number of partial streams in such a way that it allows reassembling the original media session in real-time at the receiving end.

The first laboratory tests together with the tests in the mobile environment have shown that the current implementation performs well and offers very low initial buffering times. More advanced laboratory tests with different latencies and throughputs between peers are still needed to highlight system bottlenecks and usability issues.

Acknowledgments

The authors would like to thank Joep van Gassel, Alex Jantunen and Marko Saukko for their valuable work as part of the development team. This work was partially supported by TEKES as part of the Future Internet program of Finnish Strategic Centre for Science, Technology and Innovation in the field of ICT (TIVIT).

References

[1] "YouTube—Broadcast Yourself," May 2009, http://www.youtube.com/.

[2] "Octoshape," May 2009, http://www.octoshape.com/.

[3] "SopCast," May 2009, http://www.sopcast.org/.

[4] J. Peltotalo, J. Harju, A. Jantunen, et al., "Peer-to-peer streaming technology survey," in *Proceedings of the 7th International Conference on Networking (ICN '08)*, pp. 342–350, April 2008.

[5] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," Internet Engineering Task Force, RFC 3550, July 2003, http://www.rfc-editor.org/rfc/rfc3550.txt.

[6] B. Cohen, "Incentives build robustness in BitTorrent," in *Proceedings of the Workshop on Economics of Peer-to-Peer Systems (P2PECON '03)*, pp. 116–121, June 2003.

[7] P. Shah and J.-F. Paris, "Peer-to-peer multimedia streaming using BitTorrent," in *Proceedings of the 26th IEEE International Performance, Computing, and Communications Conference (IPCC '07)*, pp. 340–347, April 2007.

[8] X. Jiang, Y. Dong, D. Xu, and B. Bhargava, "GnuStream: a P2P media streaming system prototype," in *Proceedings of the International Conference on Multimedia and Expo (ICME '03)*, pp. 325–328, July 2003.

[9] X. Zhang, J. Liu, B. Li, and T.-S. P. Yum, "CoolStreaming/DONet: a data-driven overlay network for peer-to-peer live media streaming," in *Proceeding of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '05)*, vol. 3, pp. 2102–2111, March 2005.

[10] N. Magharei and R. Rejaie, "PRIME: peer-to-peer receiver-drIven MEsh-based streaming," *Proceedings of the 26th IEEE International Conference on Computer Communications (INFOCOM '07)*, pp. 1415–1423, May 2007.

[11] D. A. Tran, K. A. Hua, and T. Do, "ZIGZAG: an efficient peer-to-peer scheme for media streaming," in *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '03)*, vol. 2, pp. 1283–1292, March 2003.

[12] J. Liang and K. Nahrstedt, "DagStream: locality aware and failure resilient peer-to-peer streaming," in *Proceedings of the 13th Annual Multimedia Computing and Networking Conference (MMCN '06)*, pp. 224–238, January 2006.

[13] J. Yu and M. Li, "CBT: a proximity-aware peer clustering system in large-scale BitTorrent-like peer-to-peer networks," *Computer Communications*, vol. 31, no. 3, pp. 591–602, 2008.

[14] J. Peltotalo, J. Harju, M. Saukko, et al., "A real-time peer-to-peer streaming system for mobile networking environment," in *Proceedings of the INFOCOM and Workshop on Mobile Video Delivery (MoVID '09)*, April 2009.

[15] H. Schulzrinne, A. Rao, and R. Lanphier, "Real Time Streaming Protocol (RTSP)," Internet Engineering Task Force, RFC 2326, April 1998, http://www.rfc-editor.org/rfc/rfc2326.txt.
[16] D. Crocker and P. Overell, "Augmented BNF for Syntax Specifications: ABNF," Internet Engineering Task Force, RFC 2234, November 1997, http://www.rfc-editor.org/rfc/rfc2234.txt.
[17] T. Berners-Lee, R. Fielding, and L. Masinter, ""Uniform Resource Identifier (URI): Generic Syntax," Internet Engineering Task Force, RFC 3986, January 2005, http://www.rfc-editor.org/rfc/rfc3986.txt.
[18] W3C, *Extensible Markup Language (XML) 1.0*, World Wide Web Consortium (W3C), 4th edition, 2006.
[19] "zlib," May 2009, http://zlib.net/.
[20] M. Handley and V. Jacobson, "SDP: Session Description Protocol," Internet Engineering Task Force, RFC 2327, April 1998, http://www.rfc-editor.org/rfc/rfc2327.txt.
[21] K. Moore, "MIME (Multipurpose Internet Mail Extensions) Part Two: Message Header Extensions for Non-ASCII Text," Internet Engineering Task Force, RFC 1522, Sepember 1993, http://www.rfc-editor.org/rfc/rfc1522.txt.
[22] "gtkmm—C++ Interfaces for GTK+ and GNOME," May 2009, http://www.gtkmm.org/.
[23] "maemomm—C++ bindings for the Maemo API," May 2009, http://maemomm.garage.maemo.org/docs/index.html.
[24] "VLC Media Player," May 2009, http://www.videolan.org/vlc/.
[25] "MPlayer—The Movie Player," May 2009, http://www.mplayerhq.hu/.
[26] "GStreamer: open source multimedia framework," May 2009, http://www.gstreamer.net/.
[27] "GNU ccRTP—GNU Telephony," May 2009, http://www.gnu.org/software/ccrtp/.
[28] "Boost C++ Libraries," May 2009, http://www.boost.org/.
[29] "The GNU oSIP Library," May 2009, http://www.gnu.org/software/osip/osip.html.
[30] "The Expat XML Parser," May 2009, http://expat.sourceforge.net/.
[31] "FFmpeg," May 2009, http://www.ffmpeg.org/.
[32] V. K. Goyal, "Multiple description coding: compression meets the network," *IEEE Signal Processing Magazine*, vol. 18, no. 5, pp. 74–93, 2001.
[33] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the H.264/AVC standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1103–1120, 2007.

PUBLICATION P7

Jani Peltotalo, Jarmo Harju, Lassi Väätämöinen, Igor D. D. Curcio, Imed Bouazizi, and Joep van Gassel, "Scalable Packet Loss Recovery for Mobile P2P Streaming," in *Proceedings of the Eigth International Conference on Wired/Wireless Internet Communications (WWIC 2010)*, LNCS 6074, Luleå, Sweden, June 1–3 2010, pp. 107–120. doi:10.1007/978-3-642-13315-2_9