Wide-area True Video-on-Demand by a Decentralized Cache-based Distribution Infrastructure

von

Dipl.-Inf. Carsten Griwodz

Geboren am 28. Juni 1968

dem

Fachbereich Informatik
TU Darmstadt
Hochschulkennziffer D17

zur Erlangung des akademischen Grades

Dr. Ing.

vorgelegte Dissertation

Vorsitz:	Prof. Dr. Peter Kammerer
Berichter:	Prof. Dr. Ralf Steinmetz
	Prof. Dr. Stavros Christodoulakis

Tag der mündlichen Prüfung: 2. Juni 2000

eingereicht
Darmstadt, den 6. April 2000

Table of Contents

1. Introduction 1

1.1 Goals of this Thesis 2
1.2 Unaddressed Related Issues 3
1.3 Structural Overview 4

2. Existing Work 5

2.1 Digital Video Distribution Architectures 5
2.1.1 Digital Video Broadcast 6
2.1.2 Digital Audio-Visual Council 7
2.1.3 Real Networks 10
2.2 Generic Videoserver Classification 12
2.3 Caching Strategies 15
2.3.1 Approaches to Distributed Caching 15
2.3.2 Building Blocks for Replacement Strategies 17
2.3.3 Co-operative Caching 19
2.3.4 Conditional and Partial Caching 20
2.4 Increasing Server Performance 21
2.5 Increasing Delivery Capacities 24

3. Gleaning 27

3.1 Refined Description of Patching 28
3.2 l-Patching 29
3.2.1 Expected Patch Stream Length 31
3.2.2 Expected Number of Active Patch Streams 32
3.2.3 Optimizing DM 33
3.2.4 Given Limits 34
3.3 Multilevel Patching 34
3.3.1 First Multicast Patch Stream 35
3.3.2 n-th Multicast Patch 38
3.4 Motivation of Gleaning for Caching Hierarchies 39
3.4.1 Design of Gleaning 40
3.4.2 Cost comparison 41
3.5 Conclusions 45

4. Protocol Suite 49

4.1 Requirements 49
4.1.1 General requirements 50
4.1.2 Generally convenient features 51
4.1.3 Specific requirements 52
4.2 Stream Control 53
4.2.1 Distributed Storage Media Command and Control 53
4.2.2 Real Time Streaming Protocol 54
4.2.3 HTTP Streaming 54
4.3 Video Streaming Protocols 54
4.3.1 Collection of Internet Approaches 55
4.3.2 Internet Quasi-Standard 56

4.4 Reliable Multicast Protocols ... 57
4.5 Selected Protocol Suite ... 58
4.5.1 LC-RTP ... 58
4.5.2 LC-RTCP ... 59
4.5.3 SDP ... 59
4.5.4 RTSP ... 62
4.6 Operation of LC-RTP and LC-RTCP ... 63
4.6.1 Design ... 63
4.6.2 LC-RTP Specification ... 66
4.6.3 LC-RTCP Specification ... 67
4.6.4 Experimental Results ... 68
4.7 Evaluation ... 72

5. Security and Copyright Protection ... 75

5.1 Secure Transmission by Encryption ... 75
5.2 Secure Transmission by Partial Corruption ... 77
5.3 Protection Features of Partial Corruption ... 80
5.4 Watermarking for Copyright Violator Tracing ... 85
5.5 Marking with the Chameleon Stream Cipher ... 87
5.6 Protection Features of Chameleon ... 89
5.7 Marking Extensions in Partial Corruption ... 90
5.8 Remark Approach to Marking and Secure Transmission ... 92

6. Simulation Modeling Basis ... 99

6.1 Topological Considerations ... 100
6.1.1 VoD Characteristics ... 100
6.1.2 Characteristics of VoD with Caching ... 101
6.1.3 Structural Simplifications ... 102
6.2 Overview of Workload Models ... 104
6.2.1 Reasons for Workload Modeling ... 104
6.2.2 Modeling for Momentary Load ... 104
6.2.3 Modeling Variations from Day to Day ... 105
6.2.4 Conclusion ... 107
6.3 Problems of Workload Models Based on the Zipf Distribution ... 107
6.4 Separation of Movie Life Cycles and User Behavior ... 110
6.4.1 Demonstration of the Population Size Effects ... 111
6.4.2 Long-term Life-Cycle modeling of Movie Life Cycles ... 112
6.4.3 Effects of User Population Size ... 113
6.5 Day-time Variations ... 115
6.6 Implementation ... 116
6.7 Evaluation ... 120

7. Simulation Results ... 123

7.1 Effects of increasing user populations ... 123
7.2 The Bandwidth Effect of Gleaning ... 128
7.3 The Hit Rate Effect of Daytime Variations ... 129
7.4 The Hit Rate Effect of Hints ... 131
7.5 The Strategy Degradation with Multilevel Clients ... 133
7.6 Simulation Wrap-Up ... 134

8. Conclusion 137

9. References 141

10. Appendices 151

10.1 Design of the Overwrite-Capable Multimedia File System 151
10.2 Protection from Collusion Attacks 151
10.2.1 Collusion Protection of the Chameleon Key, 2-party Identification 151
10.2.2 Collusion Protection of the Chameleon key, 3-party Collusion Attack 152
10.3 Cost Calculations for Distribution in Binary Trees 152
10.4 Analytical Distribution Model - Binary Tree 153
10.4.1 Unicast: No Patching, No Caching 153
10.4.2 Unicast: No Patching, Caching 154
10.4.3 Greedy Patching with central server 155
10.4.4 Patching with limited buffer and central server 156
10.4.5 Patching with Caching 157
10.5 Simulation Details 158
10.5.1 The Hit Rate Effect of Daytime Variations: Increasing numbers of movies 158
10.5.2 The Hit Rate Effect of Daytime Variations: Random sinus wave 162
10.5.3 The Hit Rate Effect of Daytime Variations: Two hot spots 167

11. Abbreviations 173

12. Trademarks 177

List of Figures

1. Introduction 1

2. Existing Work 5

Figure 1: DAVIC Reference Points and Interface Template.....9
Figure 2: Parts of the DAVIC specification 10
Figure 3: Pull and push server models 12
Figure 4: Media server architecture.....13
Figure 5: Media server's arrangement options.....14
Figure 6: External content retrieval options 15
Figure 7: Autonomous Caching.....16
Figure 8: Hierarchical Caching 16
Figure 9: Co-operative Caching 16

3. Gleaning 27

Figure 10: Buffer Usage in patching 28
Figure 11: Hints that patching window size may have an optimum 30
Figure 12: Patching with Cyclic Restart.....32
Figure 13: Expected start time intervals for active streams at time t 32
Figure 14: Stream setup example with first multicast patch 35
Figure 15: Request at time 36
Figure 16: Request at time 37
Figure 17: Binary tree model for the optimization of total networking and storage cost42

4. Protocol Suite 49

Figure 18: LC-RTP Communication 59
Figure 19: SDP specification for an initial LC-RTP stream.....61
Figure 20: SDP specification for an joining LC-RTP streams 61
Figure 21: Pass-through SDP specification moving from the proxy cache to the client.62
Figure 22: LC-RTP byte count supports retransmission 64
Figure 23: RTP header ([RFC1889]).....67
Figure 24: RTP header extension ([RFC1889]) 67
Figure 25: Application defined RTCP packet ([RFC1889]) 68

5. Security and Copyright Protection 75

Figure 26: Operation modi of block ciphers (cf. [Kun98]) 76
Figure 27: Distribution System for the Partial Corruption approach 79
Figure 28: Original JPEG image and the same image after the corruption of randomly chosen 1% of all bytes 82
Figure 29: Byte value frequencies in original movie and after fixed corruption 83
Figure 30: Byte value frequencies in original movie and after statistical corruption.....84
Figure 31: The Chameleon algorithm.....87
Figure 31: The extended Chameleon algorithm 88
Figure 32: Partial application of Chameleon for movies.....89
Figure 33: Random error insertion in Partial Corruption 91
Figure 34: Content Preparation in Remark.....93
Figure 35: Distribution System in the Remark.....94

Figure 36: Decryption in the combined approach 95

6. Simulation Modeling Basis 99

Figure 37: Distribution System Simplification....103
Figure 38: Day-to-day relevance change measurements....106
Figure 39: Rental probabilities compared with the Zipf distribution....108
Figure 40: Illustration of Zipf neighbour change vs. coefficient of variation in hits. ...109
Figure 41: Deviation of experimental from predicted order of popularity....109
Figure 42: Comparison of predicted and observed ranks....110
Figure 43: Comparison of rental store (left) and magazine numbers (right)....111
Figure 44: Aging according to RP(t)....113
Figure 45: Parameters derived from data 113
Figure 46: Smoothing effects of growing user populations 114
Figure 47: Rental probabilities compared with RP curves....115
Figure 48: Movie Selection Process....117
Figure 49: Simulation Class Hierarchy 119

7. Simulation Results....123

Figure 50: Effects of caching strategies on user hit rates and throughput....125
Figure 51: Decrease in hit ratio (a) and increase in uplink usage (b) with increasing number of available movies, uplink capacity 155 MBit/s 127
Figure 52: Decrease in hit ratio (a) and increase in uplink usage (b) with increasing number of available movies, uplink capacity 622 MBit/s 128
Figure 53: Refusal probabilities depending on user hit rates 129
Figure 54: Development of hit ratio (a) and of uplink usage (b) with increasing number of users, daytime variations with random peeks 130
Figure 55: Development of hit ratio (a) and of uplink usage (b) with increasing number of users, daytime variations with two hot spots peeks....131
Figure 56: HitRatio development with growing number of users under with Hints and ECT. ECT log size per movie is 8 (a) and 64 (b), respectively. 132
Figure 57: Multilevel clients 133

8. Conclusion....137

9. References....141

10. Appendices....151

Figure 58: Parameter setting for 10.5.1....158
Figure 59: Node descriptions for 10.5.1....159
Figure 60: Network descriptions for 10.5.1....159
Figure 61: Topology for 10.5.1 160
Figure 62: Decrease in HitRatio with increasing number of available movies, uplink capacity 155 MBit/s....160
Figure 63: Uplink requirements increase with number of available movies, uplink capacity 155 MBit/s 161
Figure 64: Uplink requirements increase with number of available movies, uplink capacity 155 MBit/s 161
Figure 65: Uplink requirements increase with number of available movies, uplink capacity 622 MBit/s 162

Figure 66: Parameter setting for 10.5.2 163
Figure 67: Hit ratio development with increasing number of users, 155 MBit/s uplink, 32 GB cache 163
Figure 68: Hit ratio development with increasing number of users, 155 MBit/s uplink, 64 GB cache 164
Figure 69: Hit ratio development with increasing number of users, 155 MBit/s uplink, 96 GB cache 164
Figure 70: Hit ratio development with increasing number of users, 155 MBit/s uplink, 128 GB cache 165
Figure 71: Throughput development with increasing number of users, 155 MBit/s uplink, 32 GB cache 165
Figure 72: Throughput development with increasing number of users, 155 MBit/s uplink, 64 GB cache 166
Figure 73: Throughput development with increasing number of users, 155 MBit/s uplink, 96 GB cache 166
Figure 74: Throughput development with increasing number of users, 155 MBit/s uplink, 128 GB cache 167
Figure 75: Parameter setting for 10.5.3 168
Figure 76: Hit ratio development with increasing number of users, 155 MBit/s uplink, 32 GB cache 168
Figure 77: Hit ratio development with increasing number of users, 155 MBit/s uplink, 64 GB cache 169
Figure 78: Hit ratio development with increasing number of users, 155 MBit/s uplink, 96 GB cache 169
Figure 79: Hit ratio development with increasing number of users, 155 MBit/s uplink, 128 GB cache 170
Figure 80: Throughput development with increasing number of users, 155 MBit/s uplink, 32 GB cache 170
Figure 81: Throughput development with increasing number of users, 155 MBit/s uplink, 64 GB cache 171
Figure 82: Throughput development with increasing number of users, 155 MBit/s uplink, 96 GB cache 171
Figure 83: Throughput development with increasing number of users, 155 MBit/s uplink, 128 GB cache 172

11. Abbreviations 173

12. Trademarks 177

How then shall I, Apprentice to the skill,
That whylome in divinest wits did raine,
Presume so high to stretch mine humble quill?
Yet now my lucklesse lot doth me constraine
Hereto perforce. But O dred Soveraine
Thus farre forth pardon, sith that choicest wit
Cannot your glorious pourtraict figure plaine
That I in colourd showes may shadow it,
And antique praises unto present persons fit.

Edmund Spenser,
The Third Booke of the *Færie Queene*

Acknowledgements

I want to thank those who helped and cared. Especially,
Prof. Ralf Steinmetz, for support, trust, and countless opportunities.
Prof. Stavros Christodoulakis, for his support and valuable advice.
My family, for all the help that I would care to take.
My colleagues at KOM, for always open doors.
My former colleagues at IBM, for keeping my curiosity alive.

1. Introduction

The world-wide distribution of multimedia documents is today restricted to discrete media content such as web pages, downloadable scripts and the rare audio or video clip. Furthermore there is an increasing number of on-line conferences and live-stream “broadcast” sources that operate at very low bit-rates. Their applicability at a world-wide scale is extremely limited because they suffer from extensive loss and delay. For in-house multimedia program distribution of multinational companies or consortia, systems exist that allow for centrally controlled distribution or a limited amount of data, and that operate on the basis of an overprovisioned, dedicated networking infrastructure. To close this gap, new and old media companies are merging at this time to bring high quality content and the Internet together. While such deals bring the infrastructure and the content together, the technology for world-wide offerings is not automatically achieved.

The initial ideas were developed at a time when many video-on-demand trials failed, which had been started when the original competitors for a video-on-demand (VoD) business were caught up in a premature hype. Their monolithic or at least centralistic architectures were developed under the approach of server-centric computing; competitors were frequently telephone and hardware companies, lacking relevant content as well as end-user infrastructure. These early developments required highly priced servers and had multiple points of failure. My assumption became that many of the problems of field trials were due to this fact specifically: centralized, costly, mainly proprietary systems as basis for distinct, non-cooperative service that provided marginally more functionality than broadcast television.

I participated in the development of the Ultimedia Server, an early product for Internet video streaming that was withdrawn soon after or even before their releases. While my interest with video streaming remained, the following projects in which I was involved, such as Globally Accessible Services (GLASS), Global Business Environment (Globe) and the EU-sponsored project Hypermedia News-on-Demand (HyNoDe), aimed at the multimedia service architectures that took video streaming as granted for a service. Personally, my opinion remained that research in wide-area multimedia distribution has not been resolved in such projects. Related research in the VoD area tended to focus on optimizing single aspects (CPU scheduling, disk scheduling, local quality-of-service guarantees, network quality-of-service guarantees, etc.), but products need to take complete system applicability into account. I did not see products that even considered large-scale delivery systems, except for proprietary distribution systems. For the large-scale distribution of multimedia content in wide-area networks for commercial use, the requirements are not fulfilled by any of the existing approaches. For commercial exploitation, the per-stream bandwidth of the in-house distribution systems would be required for an attractive quality of the continuous media content. Maintenance of the system must be reasonable, which prevents central administration of a truly large system. The necessity of quick expansion is another factor that inhibits central controls, since uninhibited growth of the system is much more realistic with a single, central control. The Internet, on the contrary, has demonstrated the benefits of decentralized organization for a continuous operation on a wide scale with its rapid growth. Furthermore, stable access to the content of a provider must be guaranteed, which implies independence from a central location and from a single network. I decided that a design needs not to consider parts, but a complete system

which must be economical, highly scalable, and without the need of a central management entity.

With the expansion of the Internet, a quickly growing number of low-quality video clips are offered to the public, while high-quality VoD is only achieved in tightly controlled in-house networks (currently called Intranets). The most prevalent software provider and integrator for low quality video-on-demand in the Internet is currently Real Networks, which has started to develop video proxy caches for live transmissions. VoD providers for the Intranet are typically offering a distribution architecture as well. In all of these cases, architectures and implementations are proprietary and, typically, centrally managed. At this time, video clips for Real Network's players and for Apple's Quicktime are the most frequently found formats on the web, which marks probably the end of the competition in the low bit-rate video market. In contrast to typical web content such as text and graphics, these approaches require streaming from the original server or manual replication to and retrieval from secondary sites. Other content is frequently stored in proxy caches which reduces latency that is experienced by the end-user and unloads wide-area networks. Because of the size of video content, especially when high-quality movies are concerned rather than the short clips that are distributed today, caching means an even more relevant reduction in the required bandwidth. Due to the decentralized organization of the Internet and its success, as well as the hardly regulated expansion of Internet services such as the web, this distribution infrastructure has demonstrated its scalability. This is an important incentive for the design of a VoD architecture in a similarly decentralized manner.

To implement such an administrative approach, it is to understand which networking infrastructure is applicable to achieve this. The introduction of new functions in the Internet has been notoriously troublesome: particularly I have been able to observe rather closely the failure of ST-II (resource-reserving Stream Protocol), of XTP (eXpress Transfer Protocol), the difficulties of IPv6 deployment in spite of its important functional upgrades, and of RSVP (ReSerVation Protocol). All of these approaches suffer from the problem that their success depends on pervasive infrastructure modifications in all parts of the Internet. My conclusion is that backward compatibility in the networking infrastructure and, if anyhow possible, in the end systems must be maintained. I assume that servers do not need backward compatibility like this since they are most probably installed as dedicated machines for the distribution system. In order to use what is available, RTP (Real-Time Transfer Protocol) is a possible choice. Because of this observation, I concluded that RTP-compliance was required.

1.1 Goals of this Thesis

The main goal of this thesis is to investigate the requirements and the options for a communication architecture that supports wide-area video distribution without central control. Although this is related to other issues, such as storage or management systems, this work focuses on the influence of decisions in the communication system.

Specifically, the thesis aims at the development of decision mechanisms for distribution and caching of high-volume multimedia contents (video objects, with volumes of 1 GB and above) and the removal of such objects from caches as they loose their value due to a reduced popularity. Although it was initially assumed that these mechanisms would be similar for other high-volume content, the requirements of real-time streaming are playing a major role in the investi-

gations of the thesis. Consumer quality is meant to imply a video quality comparable to video cassettes from movie rental stores, which provide a service that is similar to public VoD.

The investigation on the proposed distribution system includes user modeling, analytical considerations and the simulation of caching strategies. It includes also an implementation of an Internet protocol suite that can be used in several variations of the distribution system. All of these elements need to be understood individually, as well as in their interaction, to identify the differences of video distribution from other distribution systems. Examples of such differences are the relatively small number of different items that are accessed in comparison to the number of expected requests, or the slow popularity change that is observed with movie titles. The connections to other aspects of the system are made as well. This includes networking QoS, as well as an efficient interaction with a cache servers' storage subsystem. Chapter 4 addresses mechanisms that will support quality of service guarantees for the delivery of single streams, when they are implemented in a distribution system.

While watermarking and encryption have found wide-spread acceptance to solve copyright issues in access to multimedia databases, research is still going on intensely. Since decentralized systems, which include those that are based on untrusted caches are rarely even addressed up to now, a new approach has been developed. This may seem rather uncorrelated with the other parts of the work. For a consistent proposition of a decentralized system, however, addressing the copyright issue is absolutely necessary.

1.2 Unaddressed Related Issues

With the focus on the data communication in the envisioned scenario, several other aspects that are concerned by wide-area video distribution are not investigated.

Information retrieval is a topic with close relation to distributed video servers research, but they are not addressed in this thesis. While the research of multimedia data distribution that originates in this area is consistent with the communication-oriented view, a lot of the information retrieval-oriented work is aimed at efficient location, retrieval, or indexing ([CAA93]). These are considered orthogonal to the issues addressed here. The long-distance access to the relevant tools and information to handle database-specific problems should be addressed by investigating the dissemination of structured information. The specific problem of this thesis is concerned with the large volume streaming that is required for true VoD.

This thesis does not deal with charging and billing issues, except for providing an approach to enforce a reporting of content retrieval actions to content providers. Considering the nature of commercial multimedia-on-demand, a variety of policies ranging from subscription to pay-per-byte may be applicable for content providers. Network providers need to improve their infrastructure with the growing resource requirements, which generates costs that are much more difficult to assign to individual customers. [KSSW00] addresses charging at the network level, which may lead to a solution for this issue. The goals of this thesis, which assume the reliance on currently available Internet technologies, are not met by research in the charging area yet, since the deployment of the necessary infrastructure will not take place in the near future.

This thesis does not deal with the synchronization of multiple continuous media streams that are part of a single multimedia presentation, either. Given the approaches guarantees men-

tioned in Chapter 4, sufficient synchronization work has been presented in the past to allow for the synchronization of continuous media streams under well-known conditions (e.g. [JG97]). The thesis takes the potential availability into account and the presented decision will not become invalid in such an environment.

1.3 Structural Overview

Chapter 2 addresses existing work in the area of video servers and distribution of content among video server. In order to provide an insight into video servers, the most relevant video-on-demand architectures are presented, and approaches to video server design are classified. The chapter presents also an analysis of existing caching strategies in various computer science fields.

Chapter 3 discusses distribution mechanisms that have been developed for optimized delivery of on-demand video from single server systems. Especially the patching/stream tapping approach is investigated more closely, extended and applied to distribution hierarchies of multiple servers. The various approaches are compared with each other.

Chapter 4 presents the requirements that are imposed on the communication mechanisms by distribution systems in general; it explains in more detail the need for a reliable multicast protocol and the requirements of protocols that are intended to operate in conjunction with caching. A protocol suite is developed and proposed as a generic solution.

Chapter 5 approaches copyright protection arguments that are presented against the applicability of wide-area video-on-demand. On the basis of earlier work, a new architectural approach for the detection of copyright violations is presented.

Chapter 6 presents the basis of the model that is used for the analysis and simulation in this study. The basis of this information is, naturally, incomplete, since no distribution system with the expected qualities has been installed yet. This chapter validates the decisions being made.

Chapter 7 investigates the effects of video caching in a variety of ways, and draws conclusions for real-world applications of decentralized caching approaches.

Chapter 8 concludes the study with an evaluation and integration of the investigated elements.

2. Existing Work

Large scale video-on-demand (VoD) systems require the arrangement of the servers that offer the video retrieval and playback services in a distributed system in order to support a large number of concurrent streams. The approaches to create such an infrastructure range from completely centralized systems that use dedicated hardware at the network layer without intermediate data storage to completely decentralized systems that replicate all content to nodes that are close to the customer.

Since this thesis is intended to present the missing links for a complete, end-to-end decentralized video distribution system, an overview of these existing approaches from the global architecture to the existing performance tuning options is presented in this chapter. The chapter focuses on the existing work in video stream delivery and not on management or applications - although other aspects like management and security are equally important to the functionality of a complete system, efficient video stream delivery is at the core of distributed video-on-demand and, of course, our work on servers and protocols is the origin of the thesis.

The chapter starts with an overview of video distribution approaches, including the important standards of the Digital Video Broadcasting (DVB) project, which is growing but which does not support true video-on-demand applications yet. This presentation of architectures is followed by a general classification of video server designs, since the designs are typically influenced by the particular origins and philosophies of video server designer. Based on this, existing video server work is classified by this model in the following section. Several options for scaling of the delivery capabilities in such servers to handle high-volume continuous media streams are discussed after this. Finally, caching in particular is discussed with respect to its use in video distribution systems.

2.1 Digital Video Distribution Architectures

A variety of video distribution architectures exists and has been implemented so far. The most generic distinction is probably between standardized architectures and proprietary architectures. The most widely spread standardized architecture for video delivery is the result of the DVB project. Although not suited for true video-on-demand up to now, DVB standards have been adopted by several countries, and the standards are continuously extended to fulfill new and upcoming requirements. New video-on-demand aspects have recently been adopted from the Digital Audio-Visual Council (DAVIC). It has been founded to develop specifications that allow interoperability between many providers of building blocks for the provision of generic interactive services. Although no DAVIC-compliant systems exist, many components are already built to have DAVIC-compliant interfaces.

The number of proprietary systems is constantly changing. At the time of this writing, the most important system is the Real System by Real Networks, Inc., which has bought a large number of its competitors. Its transport system and codecs are largely based on research work of the recent years. Their approach is largely oriented at that of the web, and new developments make it rather similar to the approach proposed in this thesis.

2.1.1 Digital Video Broadcast

The Digital Video Broadcasting Project was founded in 1993 as the result of the work of the European Launching Group, which brought together broadcasters, consumer electronics manufacturers and regulatory bodies to prepare the development of digital terrestrial TV in Europe. The foundation of DVB sped up developments towards digital television that were already going on, such as the development of HDTV (high definition television).

The DVB interests expanded to the satellite broadcast and cable areas, providing a basis for a unified development in all relevant delivery techniques. In time, the latter areas become the better known aspects of the DVB works because the problems to be solved were, on the one hand, easier to solve and, on the other hand, more pressingly required.

In 1997 the initial work of the DVB project was considered done, and the promotion of the results through standardization bodies (in addition to actual implementations) began - DVB results are now standardized by ETSI, the European Telecommunication Standardization Institute. In the meantime, DVB has attracted the interests of more than 30 countries world-wide and DVB-compliant equipment is widely available. DVB-compliant television services are already operational, mainly in Europe, East Asia and Australia ([h:Dvb98]).

Particular standards

The video and audio encoding of the DVB standards is no re-development; rather, MPEG-2 has been selected as a coding standard and has also been influenced by DVB requirements. MPEG-2 leaves some flexibility for the application area to define, which allows other standards some configurations, such as the programme tables added by DVB. For the development of this content over a variety of media, the project created a set of standards:

- DVB-S is the specification of a satellite distribution system that can be used with any current or future transponder,
- DVB-C is a specification of cable distribution, also applicable to existing installations,
- DVB-T specifies a digital terrestrial system,
- DVB-MC/S specifies a microwave multi-point video distribution system,
- DVB-SI adds a service information system, which allows navigation information and is currently used in installations for basic program information,
- DVB-CA is a common scrambling system,
- DVB-CI defines a common interface for conditional access, e.g. for use with SmartCards

The DVB Design Goals

- Openess

The standards developed by DVB are available to everybody world-wide for a nominal fee from ETSI. To achieve not only freely available standards but also applied standards, the development is achieved by consensus in the individual working groups.

- Interoperability

Interoperability of several producers' products has been demonstrated frequently by the members of DVB from the early phases of the project. On the coding level, the wide acceptance of

MPEG-2 for most high-quality systems is supportive of this goal; on the signalling level, the work of DVB has achieved this goal.

- Flexibility

The flexibility of DVB is mainly due to the flexibility of MPEG-2: including support for high quality HDTV content, support for multiple standards such as PAL and NTSC, or the inclusion of content data (such as MHEG in the UK system).

- Market-Orientation

The participation of many vendors in DVB and the consideration of commercial requirements, and also the time scale of the development, which kept pace with the market development, have made DVB successful so far.

Relevance to this thesis

From the introduction of the DVB project above, its focus on broadcasting is clear, so it is out of scope for the objectives of this thesis at this time. First of all, DVB is in place today, and is developing towards the support for interactive services.

The implementation of MHEG object carousels in the UK installation[1] is one step in this direction, the adoption of DAVIC back-channel mechanisms for a next step in bi-directional communication is another. Several other requirements such as the support for multiple, provider-specific systems for conditional access exist in the specification and need to be considered as requirements also for a decentralized system. DVB systems show also the limits of the system: e.g. in spite of the standardization work, the integration of multiple content providers' programs into a single MPEG-2 broadcast stream has proven to be a complex task due to the centralized nature of the DVB-SI service specification.

2.1.2 Digital Audio-Visual Council

The Digital Audio-Visual Council (DAVIC) has been founded in 1994 as a non-profit Association to specify open interfaces and protocols for interactive digital audio-visual applications and services, which means basically, video-on-demand and additional services.

After its foundation, DAVIC attracted immediately a large amount of interest from industry and research; certainly this included a lot of amount of market-driven interest, since large-scale video-on-demand was expected to expand quickly at that time. DAVIC has worked by publishing calls for proposals in all technical and administrative areas that are affected by developments in the interactive digital media market. These have been answered both by industry and research organizations. Starting from a course-grained architecture that assigns reference points between service layers and between cooperating entities, DAVIC has tried to specify the details of all of these reference points. The basic philosophy, which has been adhered to in large parts, is expressed by the motto "One functionality, one tool" ([h:Davi98]). It demands that only a single standard or quasi-standard is chosen for each reference point.

1. http://www.bbc.co.uk/digital

The DAVIC Design Goals

DAVIC was founded “with the aim of promoting the success of interactive digital audio-visual applications and services by promulgating specifications of open interfaces and protocols that maximize interoperability, not only across geographical boundaries but also across diverse applications, services and industries” ([h:Davi98]).

The aspects of work have an extremely wide range, starting with hardware interoperability and aimed at the interoperability of distributed applications. The reason for this breadth is the goal of providing users with an integrated access to all information and communication, and the goal that service providers can offer transport based on hardware by several competing hardware providers and software by software providers, and content by content providers - to “support unrestricted production, flow and use of information”. The particular design goals have been defined to overcome the limitations of standardization:

- Not systems but tools

The DAVIC results are intended to be independent of specific systems; the approach for independence is the collection of target systems, their analysis, segregation into components, and the specification of these components. These are then considered the “tools” to build the complete systems.

- Relocation of tools

The first design goal specifies components that are usable in multiple systems. However, under the assumption that such components may solve tasks in several parts of a system, they should also be specified in such a way that they are relocatable inside one system.

- One functionality - one tool

Components should be unique, which is usually hard to achieve, especially since the group is receiving contributions from many industrial contributors with commercial interests. However, the goal is important; it makes interoperability and the development of complete systems easier.

- Specify the minimum

The DAVIC specifications should not include requirements that are convenient rather than absolutely necessary. They aim at world-wide applicability in a large number of fields. By permitting specialities of one group of participants into the specification, the attractiveness to other groups may be reduced. So, like the previous goal, achievement of this goals makes interoperability and completeness possible.

After the finalization of specification 1.4, ubiquitous television and the full integration of IP-based systems into the specifications are the next goals of DAVIC - the goal is to bring the advantages of multimedia in the Internet to the typical TV user, as well as to add the Internet as another distribution network to the set of specified networks such as satellite or cable.

Particular Standards

The goal of interoperability was approached by DAVIC by the definition of a “reference points and interface template” (Figure 1), which is one of the most frequently used figures of the

specification. It is intended to all the partitioning of all DAVIC applications into service layers that provide a standardized service and communicate by standardized interfaces.

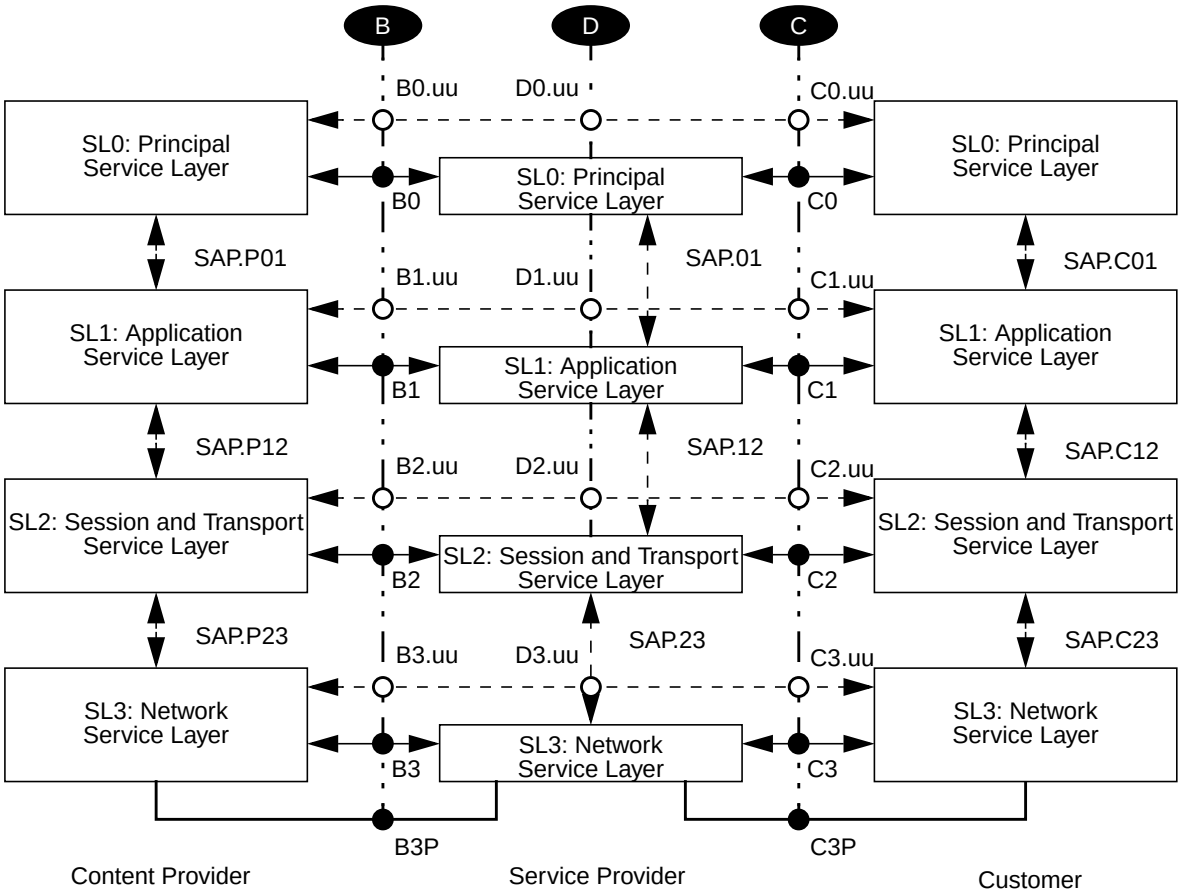


Figure 1: DAVIC Reference Points and Interface Template

In Figure 1, DAVIC intends to specify SAPs (service access points) between SLs (service layers), and peer-to-peer interfaces at each level. The peer-to-peer interfaces are specified at each level (0-4), where B labels interfaces between content provider and service provider, C labels interfaces between service provider and customer, and D labels logical interfaces between content provider and customer. Three XY.uu interfaces at each level indicate that the communication between content provider and customer must also be supported by the service provider. The X3P interfaces specify the communication at the physical layer. To standardize all of these interfaces, the DAVIC 1.4 specification consists of 14 parts (Figure 2).

Relevance to this thesis

DAVIC works by refinement: starting with contributions of intended applications, a superset of functionalities was created. The motto “One functionality, one tool” was frequently observed as an inhibiting factor for the completion of the whole from its items. The motto in itself is plausible, as is the approach of calling on existing solutions for each reference point. But to do this in independent working groups opens the door to unfruitful decisions, like extracting a tool for one functionality from one self-contained toolset, and extracting another tool for another functionality from a competing self-contained toolset. Examples at the application level are the

difficult combinations of MHEG and Java, and the support of MPEG stills (mutually exclusive with MPEG video) and DAVIC's own CLUT bitmap format.

Many companies have enhanced their existing products by DAVIC-compliant APIs, however completely-compliant DAVIC systems are very rare; it is more likely that the DAVIC work will be slowly assimilated by organizations such as DVB, which are slowly expanding into DAVIC's target application area.

Part 1: Description of Digital Audio-Visual Functionalities
Part 2: System Reference Models and Scenarios
Part 3: Service Provider System Architecture
Part 4: Delivery System Architecture and Interfaces
Part 5: Service Consumer System Architecture
Part 6: Management Architecture and Protocols
Part 7: High And Mid-Layer Protocols
Part 8: Lower-Layer Protocols and Physical Interfaces
Part 9: Information Representation
Part 10: Basic Security Tools
Part 11: Usage Information Protocols
Part 12: System Dynamics, Scenarios and Protocol Requirements
Part 13: Conformance and Interoperability
Part 14: Contours: Technology Domain

Figure 2: Parts of the DAVIC specification

2.1.3 Real Networks

Real Networks' RealSystem is the best known proprietary cross-platform video solution, built on top of UDP. It is composed of the server, RealServer, a originally platform-independent client application named RealPlayer (now only for Windows and Macintosh), and a set of proprietary encoding formats, collectively called RealMedia, as well as some other tools. Real Networks has recently begun to offer RealProxy, a variation of the RealServer that allows the creation of an architecture which is very similar to the one that we are envisioning.

Target User Group

With its system, Real Networks aims at two markets:

- Initially, the target were web site owners in the Internet, who might be interested in providing video clients on demand to their users. At first, this applied mainly to news agencies, but subsequently many others sites make use of low quality streaming video clips for extended advertisements.
- With the introduction of the G2 system, the company has entered (and largely taken over) the in-house video distribution market.

The success of the system in the Internet was due to the fact that the Real System 5.0 codecs RealVideo and RealVideo Fractal (originally ClearVideo from Iterated Systems, Inc.) in con-

junction with the company's proprietary transport system provided a working low-end video with bandwidths of 28.8 Kb/s and 56 Kb/s (modem speeds) and were very resistant to loss.

The G2 system copies the ideas of predecessors and competitors to add applications that are requested by the in-house market. These are largely based on standards and quasi-standards and allow the use of a large suite of third party tools that are marketed by Real Networks together with their products. Multimedia presentations integrate video, audio and text into a synchronized presentation, where synchronization is specified by the W3C-driven Synchronized Multimedia Integration Language (SMIL). XML-compliant new data types for streaming text (e.g. subtitling) and slide shows can be expected to work with many future tools. Additionally, a variety of additional third party video codecs has been added to support higher video bit rates and to achieve better video quality at lower bit rates. This include high-quality MPEG, the Windows standard format AVI, Microsoft's packaging format ASF, and VivoActive's low-end encoding. Additionally, the G2 system can be expected to scale a lot better to large sites and world-wide institutions than earlier version due to the introduction of a proxy cache server ([h:Real99]).

Specific Techniques

The success of Real System 5.0 has been the free availability of the cross-platform players, of content in this format, and of course its scalable delivery capabilities. These features are proprietary concerning both the transport protocols and the encoding formats. The Real System 5.0 supported only Real Network's own codecs: Real Video is a codec that works over low bit rates and is extremely loss-resistant. This was aimed at modem users and at the Internet with its frequent packet losses. Real Video Fractal is a fractal-based encoder, recommended for high bandwidth and frame rate applications where packet losses are expected to be low, such as corporate intranets.[2]

Relevant information for video delivery over networks with quality of service guarantees at the network level can be extracted from the decisions that were made by Real Networks in their transition from the 5.0 system to the G2 system. Originally, Real had employed a codec which implements a technique named "stream thinning", which is frame dropping due to loss detection. In spite of their own or other loss-resistant codecs, they have been working on the networking protocols to achieve more flexibility for (application level) re-negotiation of the quality, and added a large number of encoding formats. They have made the observation -in real-world applications with customer feedback rather than in experiments- that it is more appropriate to change an ongoing delivery from one encoding format to another rather than relying on the loss resistance or scalable features of a codec. The basis for this is a proprietary transport system named SureStream, which runs on top of UDP unicast or multicast and provides feedback to the sender.

The other important new addition to the Real System suite is the Real Proxy, a proxy cache that works independently of central control. It can be installed by site administrators as a proxy server, or it can be configured in the Real Player preference settings by the user. It authenticates the user's request to the server and determines bandwidth requirements. However, it

2. It is interesting to see that this information is released only after the acquisition of better codecs from other companies

stores only live streams in expectation of additional viewer who may connect to the same live stream later. Other content is only passed through.

Relevance to this Thesis

Real System as the most important video-on-demand system on the Internet has always been an important VoD approach to observe - in spite of being used only for video clips. Since the introduction of the proxy server, the system is interesting to evaluate against the goals of this thesis. In some design decisions the importance of the Real System was the reason for our choice of tools, e.g. without the effort of the Real Networks in standardizing RTSP, we would probably not have adopted this for our protocol suite (Chapter 4) but would have followed one of the many straight-forward approaches of existing research prototypes or we perhaps DAVIC's proposal DSM-CC.

Design details that we have considered necessary have independently been implemented in this server, like reporting to the content provider's server (Chapter 5). Other aspects show that the Real system is a straight-forward extension of existing tools and techniques, e.g. it applies a true write-through approach, which performs much better than the store-and-forward approach of web proxies, but which implies redirection of all data streams from the central server through the proxy, which we consider a waste of bandwidth (Chapter 4).

2.2 Generic Videoserver Classification

Videoservers are a special variation of file servers with the requirement to deliver part of all of their services within a certain time-frame. This basic requirement can be addressed at more than one of the hardware and software levels which comprise a media server. Consequently, the range of research issues that contribute to media server design is wide. While many research groups deal with multimedia servers as a database issue, this chapter of the book concentrates on multimedia servers' content storage and movement and does not consider its management.

A basic, application-specific distinction is made in these design of such media servers: data retrieval can either be controlled strictly by the client, which requests and sends pieces of content files, or a client can tune in to a server-controlled sending of data, which might have been initiated by that client. Figure 3 demonstrates the request/response behavior of both approaches.

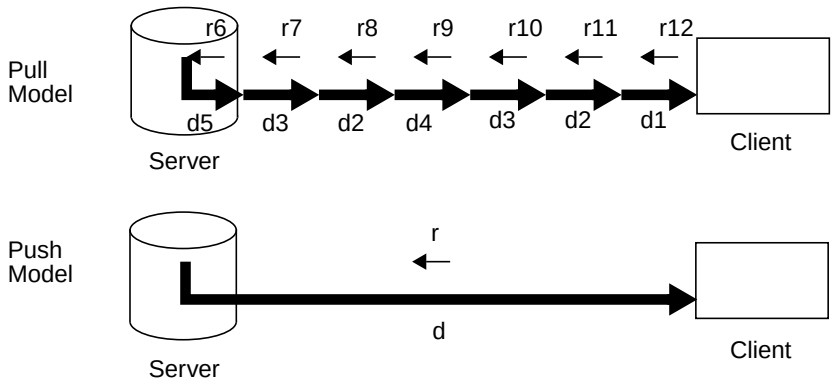


Figure 3: Pull and push server models

A media server that is operated in the first mode is called a *pull server*, a server that is operated in the second mode is call a *push server*. Another, frequently used expression for the push server is the term *data pump*, as this characterizes in a simple way its specialization in retrieving data from disk and delivering it to the network efficiently. Pull servers are surely the more appropriate choice for editing multimedia content in a LAN environment: linear retrieval is frequent but not the rule, pieces of content are rearranged, temporal and spatial cross-connections are introduced. Push servers are the obvious choice for broadcast or multicast distribution of content over wide areas, with no or infrequent user interaction. Applications that are not as clear-cut in there requirements may be solvable with either of the two approaches.

Pull and push servers are often considered competing concepts. Media server implementations, however, show that these worlds are not far apart from each other because major parts of a server can be operated in modes that can be used in pull as well as push mode. So-called *Play lists* mix these two modes and are one real-world solution to existing application requirements. Such *play lists* are client-defined lists that refer to pieces of content that are stored on the server; the pieces indicated by play lists are supposed to be sent to the client in a sequence [RFC2326]. In the following, we do not separate these two approaches any more.

Media servers are responsible for the timely delivery of content to an end-system. To achieve this goal, each component of the media server must conform to the bounds of time and space to fulfil its tasks. This attracts the research in a variety of areas: disk layout strategies, disk scheduling, file systems, data placement, memory management or CPU scheduling. Figure 4 shows the order in which media server components are involved in delivering the content. Some of the tasks that are separated in that figure are historically implemented in a single system component.

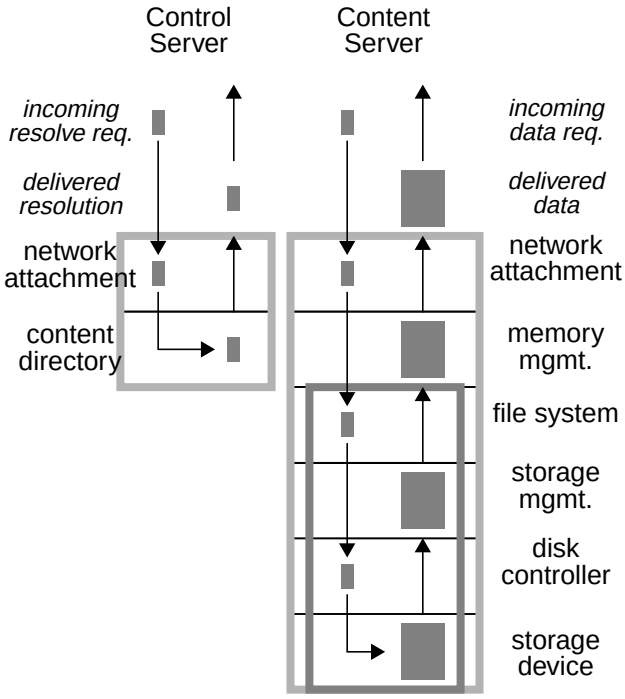


Figure 4: Media server architecture

The *network attachment* is typically a network adapter or a similar device that connects the media server to the customers. The *content directory* is the entity responsible for verifying whether content is available on the media server and whether the requesting client is allowed to access the data. The *memory management* is a separate entity because although a typical content file of multimedia applications is too large to be kept in the main memory for a long time, the caching of content data in main memory improves the performance considerably for some applications. The *file system* handles all information concerning the organization of the content on the media server. This includes such issues as the assignment of sufficient storage space during the upload phase, probably the transparent segmentation of the content file, the consistency of the data on disk, and the loca-

tion of the elements of a segmented content file during retrieval operations. The *storage management* is the abstraction of driver implementations that communicate directly with the disk controller. The storage management is concerned with disk scheduling policies and the layout of files. The *disk controller* handles the access to data on the *storage device*. Research on the disk controller level includes the increase of head movement speed, I/O bandwidth, the largest and smallest units that can be read at a time and the granularity of addressing.

Of course, optimizing one of the components is not sufficient. The components must cooperate correctly even when the system grows. Such a growth means that the system or some of its components will be replaced or extended. In many cases an extensions means that a task is distributed onto multiple components, probably onto heterogeneous components, and that it may become necessary to replicate part of the data to access it from all components of the distributed system. [TKS94] provides a formalization of the options for distributing parts of a video server. This formalization deviates from the reality with the generalization of the content directory's position in a distributed system. In a typical video server, the content directory should always be complete and consistent, in order to answer requests correctly. Figure 5 demonstrates the two alternative approaches to generalizing component distribution while a consistent content directory is maintained. Figure 5 (a) uses an internal content directory which, for

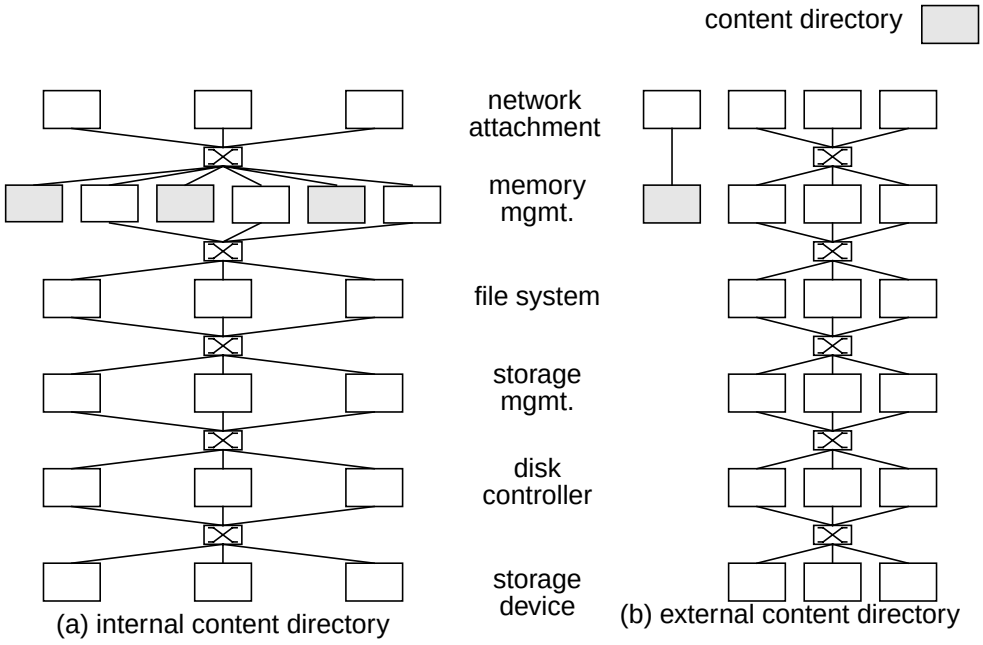
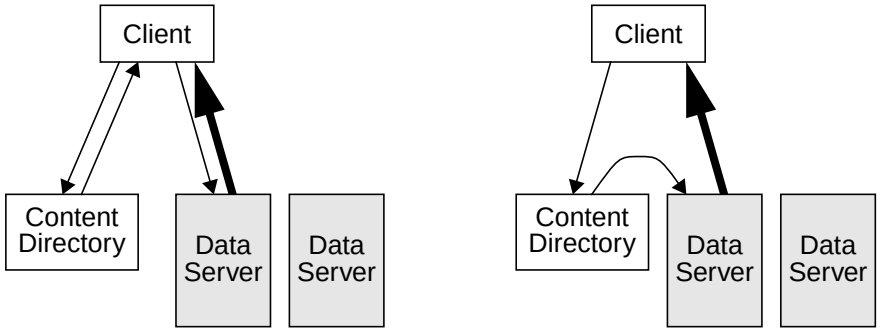


Figure 5: Media server's arrangement options

consistency reasons, can exist only once per media server. However, although the content directory appears consistent to all other components, it may still be distributed internally and achieve the appearance of a single component by presenting the same interface on all nodes of the media server. Figure 5 (b) shows all options for distributing components when the approach of an external content directory is adopted. A client of such a system contacts the external content server first to identify itself and to issue the request. After that initial request, two alternatives for proceeding with the retrieval operation are possible. If the response of the content server is returned to the client and the client is responsible for issuing the actual request for

data in another call (Figure 6 (a)), additional security mechanisms must be applied because authentication of the client is checked by the content server. Alternatively, the content server can accept all requests directed to the media server, but instead of answering itself, it can immediately order the appropriate nodes of the media server to deliver the content data (Figure 6 (b)). This approach is restricted because it requires one of two things: either the client must be able to receive the content data from a different server than the target of its request, or each server node must deliver the content using the address of the content server.



(a) two-step retrieval (a) request redirection

Figure 6: External content retrieval options

2.3 Caching Strategies

A caching strategy is the chosen approach of a system for the decision whether an object should be kept in a certain cache or whether it should be removed. Caching has been applied widely throughout the computer systems development, ranging from CPU's on-chip caches to the caches of distributed file systems and the web.

The primary task of caches is to decrease the latency in accessing objects by eliminating a more time- or resource-consuming communication of the accessing entity with the original location of the object.

Available cache space is typically smaller than original space. This is often, but not necessarily, based on a higher price of the caching space w.r.t. to the original space. In some cases, other reasons prevent the work from taking place exclusively on the more easily accessible storage. E.g. in case of the AFS, operations such as synchronization and persistence are bound to the original storage system, while other operations are not restricted in such a way.

2.3.1 Approaches to Distributed Caching

In wide-area distribution caches are frequently required to communicate among each other to fulfil a client request. The best-known example is the WWW, which is also a playground for the implementation of a variety of distribution system approaches for low-volume data. A general distinction of caching approaches into three kinds is possible:

- autonomous caching
- hierarchical caching
- cooperative caching

These kinds are distinct from each other concerning the co-ordination effort that is necessary to implement the overall distribution system, as well as the reliability of the system.

Autonomous Caching

Autonomous caching strategies are the easiest to implement since they do not require any communication with other caches. They are extremely resistant to networking problems since each caching node can act independently from the others.

This independence is also the reason for the major drawback of autonomous caches. In a large system of communicating caches, object requests among caches have the effect that copies of the same object will often be located in various caches between a client and the original location, while other objects are not cached at all.

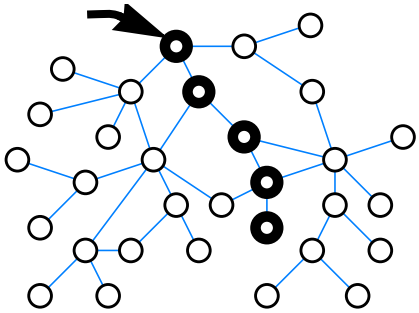


Figure 7: Autonomous Caching

Hierarchical Caching

Hierarchical caching strategies promote a centralized approach with a single node that controls the location and the movement of cached items. This approach is frequently applied in environments with very limited number of cache servers (e.g. a company-wide mail system), or in systems where functionally identical caches and hierarchically connected caches are assigned complementary tasks (e.g. in 1st-3rd level CPU caches). In wide-area distribution systems, this approach has currently problems with scalability and management. It is endangered by its reliance on a central coordinating instance that must be reachable within a limited time frame, and which must be kept aware of the status of all connected caches. In the Internet, typical examples of such hierarchical systems are the so-called push-channels [h:MaCa98] and web satellite distribution [RoBi98].

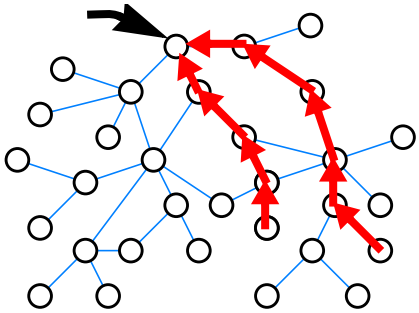


Figure 8: Hierarchical Caching

Co-operative Caching

Co-operative caching strategies are strategies that include the information from a group of caches and that make decisions for all members of the group based on the collective information.

There is no established segmentation of co-operative strategies yet since there are too few and these are too different to categorize. Furthermore, several co-operative strategies are only word-of-mouth but still undocumented in research papers.

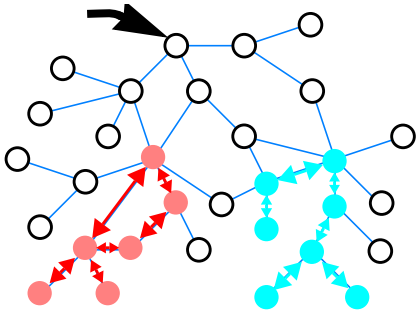


Figure 9: Co-operative Caching

2.3.2 Building Blocks for Replacement Strategies

As a further analysis of caching approaches, a large number from areas such as web caching, operation systems (paging and swapping) and distributed file systems was analyzed in [KüGr98]. One of the main results of this work was the identification of the elementary properties of cached objects that are taken into account in the removal decisions of caching algorithms. Since these properties are the building blocks for basic replacement strategies, they are explained here in detail, rather than listing a series of existing caching strategies.

Two aspects are considered separately in later sections to account for their importance. Cooperative caching (Section 2.3.3) is rarely found but has been shown to be very efficient in wide-area distributed architectures in [FJL+97]. Conditional replacement (Section 2.3.4) is rarely found as well, and existing approaches make rarely use of it: it allows a strategy to decide whether an item is stored in addition to being forwarded to the requesting user or whether it is only forwarded but not cached. A typical use is the option for web proxy caches to ignore large items entirely to save cache space.

This chapter does not take into account thtat many features of a video distribution system simplify the conditions that the caching techniques encounter for video content. First of all, video content is stored as write-once-read-many content (which does not apply e.g. to paging techniques), transfer times are extremely high, and the ratio of cache size and object size is expected to be small (both in contrast to e.g. web content).

Fair Share

This is an approach rather than an attribute, which allocates part of the available space for each object that is requested (in a “fair” way). Objects will typically not fit into the allocated space completely, but only the initial of most recently used part is stored.

Although other attributes can be used to distribute shares of different sizes based on additional relevance information, the partial storage of objects is not considered in this thesis. Although several approaches for partial caching exist (REFS), I consider them inapplicable for distribution on a world-wide scale at this time. Although router providers are starting to implement resource management-supporting techniques such as Differentiated Services ([RFC2475]), I assume that the deployment of such techniques will take longer than the rise of wide-area video-on-demand. Before their deployment, however, the completely cached objects will be preferrable over partially cached objects due to the higher average availability of complete movies.

Age

Objects have an order of being loaded into the cache. A replacement strategy can make use of this information when it looks for an object to be expunged from the cache. This attribute of an object is used as a stand-alone criterion by one of the simplest replacement strategies: FIFO (first-in-first-out).

Number of Requests

This attribute of cached objects provides a means of evaluating an object’s popularity. Since it is not tied with any concept of time in its basic form, this popularity can keep very old objects

in the cache that have been entirely unpopular recently. Because of this problem, the unmodified use of this condition requires that request counters are reset. A typical application is the use of request counters with a common reset interval (each day, each week) for all objects in the cache.

Aging

Aging is not an attribute in itself, but an application-dependent technique that modifies attributes (e.g. the number of requests). The techniques are used to give younger requests a higher relevance for removal decisions than older requests. Consequently, a removal strategy that applies aging can adapt to changes in objects' popularity.

Intervals

Intervals are used to implement request histories for items that is less complex than aging. Rather than computing weights for consecutive requests, measured attributes are collected for the time of an interval and than simply discarded. This prevents errors that may otherwise persist in the decision-making process for a long time, but immediately after the clearing of the attribute cache, decisions are mainly random.

Time since Last Request

The time since last request can be used together with a simple sorting algorithm to identify the object that has been accessed most recently or least recently. By itself, it is used in the LRU (least-recently-used) strategy which is a frequently applied and simple, requires only very little storage space and computing power and yields not-too-bad results in many cases.

Size

Since the available space of a cache is always limited, it can be filled up with a small number of large objects. In many cases, this is less advantageous than the caching of a large number of small objects, which increases the average response time of all user requests. This is the reason for web cache administrators to limit the size of objects that are cache and to refuse the caching of large object altogether.

Cost of Transfer

This is a comparison of the costs for keeping an object in a cache with the costs for transferring an object from a different server on demand. The comparison is of major importance in situations where objects are requested from original servers with differing access costs. In the simple case, transfer costs are measured in transfer time, but in other situations, there may be charges for using a link.

Cost of Transfer and Storage

This is a more complex variation of the cost for transfer, where keeping an object in the cache may be generate costs, e.g. due to charges of the copyright holder for storing a copy of an object locally, or charges of a storage maintainer for renting storage space.

Bandwidth Usage

The variations of bandwidth of links among caches are a relevant part of decisions in replacing objects.

Cache Cleaning

If caches are cleaned regularly, the time that unpopular data can remain in the cache has a fixed limit. On the other hand, popular data is removed from the cache as well and has to be retrieved. It may have some validity with co-operative caches that hold object replica.

Priorities

Priorities can be used to move objects through a distribution system quickly if popularity or relevance can be predicted. This is the case for heavily advertised videos that are supposed to be available to all users at a certain time. In case of large objects, large object transfer times, and short popularity cycles[3], priorities may be used to keep objects explicitly through non-popular times of a cycle.

2.3.3 Co-operative Caching

Hint-based Caching

Co-operative caches can exchange information with other caches in a group. These hints are used by a cache that experiences a cache miss to find objects in caches that are close to itself and potentially closer than any server between the cache and the origin of the requested object. The removal decisions remain the same as before for each cache. Tewari has demonstrated that this approach can be very efficient for wide-area distribution of web content, even web content of a kind that includes video clips ([FJL+97]).

Hint-based Push-Caching

Push-caching is an ‘in’ word for pre-distribution of objects. It is an extension of hint-based caching that makes use of the requests that were experienced by multiple caches collectively. Based on this information, the objects are distributed to caches in a distribution hierarchy according to their overall hit rates. However, this approach is concerned with distribution mechanisms only.

Broad-/Multicast Push Cashing

The multicast variation of the push caching approach assumes that a sender at the core of a distribution structure (core server) decides which data is most relevant for the caches in the hierarchical distribution system. One advantage of the approach is that the core server can collect request data from a large number of caches and predict future developments more reliably. Another are the potential bandwidth savings due to the early presence of content with rising

3. It is conceivable that caches and bandwidth are too small for a full cache replacement during one cycle. Assume children’s programs should be kept in a tiny cache for the following day.

popularity, and the ability to perform bulk transmission at a predefined throughput, which allows a better prediction of network allocation. ([RoBi98]).

However, there are several problems to this approach as well, if it is not applied on web caches with large disks and a fallback connection to the Internet. Since the hierarchy is strictly two-step (core server and cache), server downtime is fatal for the system; the predicted network allocation will probably not allow the retransmission of content that could not be transmitted during the failure.

If cache sizes are heterogeneous and the caches are not extremely large with respect to the number and size of the stored items, the replacement strategy is missing entirely from the approach. Since cache servers are unaware of global hit rates for objects (and do not receive hints from the core server), they must remove objects from the cache based on their own strategy. However, popularity variations for small user populations can vary widely from the average popularity that is experienced by the core server.

2.3.4 Conditional and Partial Caching

Unconditional Overwrite

The unconditional overwrite strategies are simpler than conditional overwrite strategies since an object that is requested from the cache is always delivered, and if it is not stored in the cache yet, it will be stored due to the answered request. Replacement decisions do not have to take any properties of the newly inserted object into account but they can decide that an object is removed from the cache because of a cached object's attributes.

Conditional Overwrite

The conditional overwrite strategies are more complex than unconditional strategies since they take attributes of a candidate object that is requested into account, and have to compare the value of this object with the value of the objects that are already stored in the cache. The difficulty is that very different attributes have to be assigned a value for this comparison.

The historical uses of caching are characterized by the necessity for downloading an object completely into the cache before delivery to the requesting user. There are several deviations from the store-and-forward approach.

Cache Windows

Continuous media objects can make use of caching to reduce access latency and to increase the output bandwidth. This has been used as a design feature for various video servers with a hierarchical structure ([Lamp98]). These designs assume that main memory operates as a cache for disks, and that disks operate as caches for third level memory such as tapes or CD-Roms. In all of these cases, delay variations (jitter) between two hierarchy levels is limited and predictable, and caching only parts of the object is kept in the cache memory.

With this scheme, a small access latency is supported by always keeping the initial part of the object in the faster cache. The output bandwidth is increased by serving potentially more clients concurrently from the cached portion of the object than can be supported by the original medium without replication.

2.4 Increasing Server Performance

This section deals with means for increasing the performance of video servers. The performance aspects have been investigated to understand the service that can be expected from a server in a video distribution system, as well as implications for the interoperability of servers with the proposed distribution mechanisms. The issues of real-time retrieval from disks for multimedia streaming have been explained in [GeCh92]. A performance increase of the servers themselves beyond single file retrieval optimization can be achieved by replication of hardware resources and content, or by appropriate assignment of the content to hardware resources. Content replication is a means to answer two issues at the storage management level: to guarantee availability in case of disk or machine failures, and to overcome limits in the number of concurrent accesses to individual titles due to throughput limitations of the hardware.

Static replication

The simplest approach to replication that can be taken is the explicit duplication of content files, by storing the file on multiple machines and providing the user with a choice of access points. This is frequently done in the Internet today: the content provider stores copies of the original version up to date on servers close to the user. Using the more elaborate options, the content is duplicated manually, and an application provides alternating copies of the file under the same name. Automatic replication of the relatively small and frequently accessed read-only system files for load balancing among file servers has been proposed in [SKK+90]. A static placement policy that uses estimated load information for the placement of video objects is proposed in [DaSi95b]. This static placement policy is complementary to the proposed replication, as it reduces, but cannot eliminate, dynamic imbalances.

Dynamic Segment Replication

Dynamic segment replication as it is introduced in [DKS95] is designed for content which is accessed read-only and which can be split into equally-sized segments of a size that is conveniently handled by the file system. Fixing segment sizes as well as choosing segments that are large in comparison to a disk block are decisions that are made to keep the implementation overhead low. Since continuous media data is delivered in linear order, a load increase on a specific segment can be used as a trigger to replicate this segment and all following segments to other disks. Such segments are considered temporary segments in contrast to the original segments, which are permanent segments. One of the major advantages of this replication policy is that it takes not only the request frequency of individual movies into account. Rather than this, the load of the disk is also considered. Specifically, the decision is made in the following way: each disk x has a pre-specified threshold for the number of concurrent read requests B_x that must be exceeded by the sum of all segments' read operations in the current read cycle of the disk (where 'cycle' means the playout time of one segment) as well as by next read cycles in order to initiate the replication algorithm.

To simplify the calculation, the read requests are considered uniformly distributed over all replicas rather than taking requests to other segments on the same disk into account. In this way, the future load in t cycles for the i-th segment is predicted as n_{i-t}/r_i where n_{i-1} is the number of viewers of segment i-t and r_i is the number of current replicas of the segment. For all

segments j ($j<t$), it is assumed that the current arrival rate n_1/r_i will be maintained in the future. If the sum of the expected load for all segments on a disk exceeds B_x, the replication is triggered. Then, the algorithm must identify a segment for replication. Since the approach replicates segments only when they are retrieved from disk because of a client request, in order not to add additional load, replication can start only when a stream starts reading a new segment. Hence, if the disk load exceeds B_x at a segment boundary crossing, we must decide whether it is desirable to replicate this segment. The segment is replicated only if the replication of this segment has the highest estimated payoff among all the segments on the disk. If the gain in replicating a different segment is considerable, a boundary crossing to that segment is awaited. The estimated payoff p_i is computed as

$$p_i = \left(\frac{1}{r_i} - \frac{1}{r_i + 1}\right) \sum_{j=0}^{i-1} n_j w^{i-j-1}$$

where w is a weighting factor. w can be chosen big to put a stronger weight on long-term predictions; this is a good selection when the load on individual segments stays similar for a relatively long time. If the load on segments is fluctuating strongly, the expectation of future behavior is unreliable and should have less relevance, expressed by a lower weight w.

Threshold-Based Dynamic Replication

The threshold-based dynamic replication introduced in [LLG98] considers whole movies rather than movie segments, and it takes all disks of the system into account to determine whether a movie should be replicated. This approach accounts for the possibility that the term ‘disk’ does not mean a single physical disk but a logical disk. For instance, such a logical disk may be an array of physical disks with a single representation to the storage management. Still, it is assumed that the media server is large and consists of many such logical disks. The service capacity in number of concurrent streams of such a disk x is called B_x, the average service capacity of all disks is called $\bar{B}$.

A replica of a movie is assumed to be stored completely on one of these disks. For each movie i of length m_i, a probability of being selected in a new request P_i as well as a request arrival rate λ must be computed from earlier requests. The replication threshold T_i is than computed as $T_i = min(p_i \lambda m_i, h\bar{B})$, where h a constant value to limit the probability of replication. For each disk x, the measured current load L_x is taken into account to compute the current available service capacity A_i for serving video i by calculating

$$A_i = \sum_{x \in R_i} (B_x - L_x)$$

where R_i is the set of disks that carry replicas of i. If $A_i<T_i$, a replication of movie i is triggered. Similarly, [LLG98] proposes a decision for discarding replications when the number of concurrent requests n_{ix} on a movie i at disk x decreases. The following condition is checked before a replica is removed:

$$\sum_{y \in R_i \backslash x} (B_y - L_y) - n_{ix} > T_i + D$$

This inequality integrates two important conditions. The inequality

$$A_i = \sum_{x \in R_i} B_x - L_x > \sum_{y \in R_i \backslash x} (B_y - L_y) - n_{ix} > T_i + D > T_i$$

implies that the replication is not triggered again immediately after a de-replication, and

$$\sum_{y \in R_i \backslash x} (B_y - L_y) - n_{ix} > T_i + D > 0$$

guarantees that all streams on disk *x* can be served from the remaining replicas. *D* is an additional threshold to reduce the probability of an oscillation between replication and de-replication further.

The approach includes also the proposal to replicate a movie from the least loaded disk to the destination disk because an overhead may be induced by an additional read operation on the source disk. For the selection of the destination disk out of the set of disks that do not yet hold a replica of the movie in question, multiple approaches are considered. The most complex one takes the number of current streams into account, but assumes that all ongoing replications are already finished and the streams are distributed onto the disks as if the replicas were already active. For the replication itself, various policies are proposed.

Injected Sequential Replication adds additional read load to one disk because it behaves like an additional client, by copying the movie at the normal play rate from the source disk to the target disk.

Piggybacked Sequential Replication is identical to the replication used in the Dynamic Segment Replication: the movie is written to the destination disk while it is delivered to one client from the same memory buffer. Since this scheme makes replication decisions for a movie always during admission control for new clients, this does not add complexity to identify the source copy of the operation. However, the copy operation is affected when VCR operations on the movie are performed.

Injected Parallel Replication use a multiple of the normal data rate of the movie to replicate the movie faster from the source disk to the destination disk. In order not to inhibit admission of new customers, this multiple of the normal data rate is limited.

Piggybacked Parallel Replication copies at the normal rate of the movie, but not only from the position of the newly admitted client. Instead, later parts of the movie are copied at the same time from the buffers which serve clients that are already viewing the movie. Obviously, this approach needs unusual low level support because data is written in parallel to different positions in a not-yet complete file.

Piggybacked and Injected Parallel Replication combines the other parallel replication approaches to replicate parts by the injected approach of the movie that would have to be copied late in a piggybacked parallel replication mode because no client is expected to view those parts in the near future.

Interval Caching Policy

[DaSi95b] introduces partial replication of multimedia files for load-balancing in multimedia systems. It is based on the observation that if there were a number of consecutive requests for

the same video, and if the blocks read in by the first request were copied to another disk, it would be possible to switch the following requests to the partial replica just created.

Generalized Interval Caching Policy

The Interval Caching Policy, proposed in [DaSi95b], exploits the movement of data through the main memory of a video server by keeping the data of such streams in memory, which are followed temporarily close by another stream of the same object. This policy is refined in [DaSi95] to take into account that the interval caching policy does not handle short files appropriately when the media server is handling a mixed workload rather than a videos.

Random Duplicated Assignment

Random Duplicated Assignment ([Kor97]) is a technique that assumes large scale servers. In its basic application, it does not take different popularities into account. The basis assumption of the technique is that any non-randomized assignment of variable bit-rate content to replicated disks will result either in loss of disk space (by applying an equal time block approach) or in performance reduction. Furthermore, a generic workload is assumed, which could result in unexpected hot spots. The approach uses all available disks fairly by distributing equally-sized blocks of the content randomly, and by storing a second copy of each block on a different randomly chosen disk.

Thus, it is resistant to single disk crashes and can be extended to arbitrary safety levels, allows simple crash recovery and addition of disks. It does not degrade when disks have different features, either. While all of this seems applicable for a real-world, long-term scenario, the specific approaches mentioned earlier could probably result in better performance for dedicated systems.

2.5 Increasing Delivery Capacities

Besides the performance increase in single servers or server clusters that was presented in the previous section, additional techniques make use of the specific user behavior and user perception in the on-demand delivery of video. The techniques below exploit limits in the user perception of waiting time and display speed, and the possible acceptance of interrupts.

Batching

Batching is an approach introduced in [DSST94] to exploit the memory bandwidth and to save disk bandwidth in media servers by defining a temporal cycles called batching windows. All requests that arrive within such a cycle are collected and, at the end of the cycle, all requests to the same content are serviced from the same file and buffer. This approach weakens the on-demand idea in comparison to the interval caching policy, but it recovers potentially large amounts of main memory because content can be discarded from the main memory immediately after playout and it will be re-loaded only after the next cycle. [DSS96] modifies this approach towards dynamic batching, which services requests as soon as a stream becomes available. Two selection policies, first come first serve (FCFS) and maximum queue length (queue length is defined by the number of user who requested that file), are compared, and FCFS is shown to be the better performing.

Piggybacking

The aggregation of streams that deliver the same content in close sequence without the use of a batching window was proposed by means of piggybacking ([GLM96]), i.e. one stream of a content file that is shortly preceding another stream of the same file should be joined with the later one. The general means to do this is an increase in the speed of the later stream and/or a decrease in the speed of the earlier stream until they join. Various strategies for joining more than a pair of streams are then investigated in detail in [GLM96].

Content Insertion

For the video-on-demand special case, [VeLi95] proposes the most radical extension of that scheme to date by offering content insertion to force larger numbers of streams into a time window which is small enough to allow the use of the piggybacking technique to join them into a single stream. Such inserted content from a content loop like an eternal advertisement show or from a continuous news show might be acceptable to the user to stay tuned. Alternatives might be a lengthening or shorting of introducing scenes of a movie. In [KVL97], it is then offered that this technique can also be used for providing a just as pragmatic and radical solution to problems such as server overload or partial server failure by diverting users into an advertisement loop or presenting other fill-in content until the problem can be fixed or until an aggregation with an action stream can be performed.

Stream Tapping/Patching

For the exploitation of multicast in true VoD systems, an idea to exploit multicast for true video-on-demand was presented in [CaLo97] under the name stream tapping in [HCS98] as patching. Since our attention was first drawn to [HCS98], the term patching is preferred in this thesis. The basic approach is the creation of a multicast group for the delivery of a video stream to a requesting end-user. If another user requests the same video shortly after the start of this transmission, he starts storing the multicast transmission in a local cache immediately. From the server, a unicast stream is then sent to this user containing the missing initial portion of the video, until the cached portion is reached. Then, the end-system uses its cache as a cyclic buffer.

This approach can be applied among servers, which distinguishes it from other variations of the NVoD technique Pyramid Broadcasting ([AWY96]). It is important for the goals of this thesis that this technique does not conflict with the use of cache servers but that the reason for cache server usage changes. The relevance of cache servers for the prevention of network overload is reduced, but their relevance for system stability and an acceptable packet loss ratio for the end-user remains. For the same reasons, cache servers need to store most frequently requested movies even in this case.

In “Exploit Multicast for True Video-on-Demand Systems”, Hua et.al. argue for a centralized server in conjunction with their scheme and client-side buffering. Their argument for a centralized approach is due to administrative difficulties of wide-area distributed architectures under central control. We agree with their assessment of administrative problems in wide-area distributed systems under central control. However, central control is necessary only if copyright problems can not be solved for distributed systems without central control. Since we

expect further work on copyright protection, and hope to give some inspiration with Chapter 2 of this thesis, we discuss this technique in more detail in Chapter 3.

3. Gleaning

The use of multicast with video-on-demand systems is only of limited use, since true video-on-demand implies that a stream is delivered to a user personally and on request. Many papers propose approaches based on *staggered broadcasting* ([DSST94]), the initial NVoD idea of retransmitting a video over a fixed number of channels with equal time offsets. Refinements that are based on client-side buffering are *stream tapping* ([CaLo97]), *pyramid broadcasting* ([VI96]), *permutation-based pyramid broadcasting* ([AWY96]), *skyscraper broadcasting* ([HS97]), *dynamic skyscraper broadcasting* ([EV98]), *harmonic broadcasting* ([JT97]), *cautious harmonic broadcasting* ([PCL98a]), *polyharmonic broadcasting* (*PHB*, [PCL98b]) and *transition patching* ([CaHu99]).

Patching ([HCS98]) was basically a re-invention of *stream tapping*, but it introduced the possibility that a movie is not transmitted at all unless a request to this movie has been made. *PHB with partial preloading*, *mayan temple broadcasting* (both [PLM99]), new merging schedules ([EVZ99]) for *piggybacking* or *dynamic broadcasting*, *catching* and *selective catching* (both [GZT99]) are variations that were presented only recently, all of which apply preloading of video parts into clients without demand. All of these techniques are based on

- a central server and broadcasting assumption
- controlled repetition of movies or movie segments (with the exception of *patching*)
- knowledge of movie popularities

Patching is an on-demand ideas that sends additional streams only on request. Consequently, it is harder to compute the best values for restart times and simplifications to internal restarts are adequate.

Our own contributions were inspired by the goals of TVoD and the idea of using caches, and are documented in ([GLZS99]) and ([GZL+00]). Most computations of this chapter are presented in those papers. After an introduction of Patching which is a refinement of the overview in the previous chapter, we present the TVoD optimization of patching in Section 3.2. We state that temporal distance between two multicast streams for one movie should not be determined by a client policy or simulation. Rather, it can be calculated by the server on a per video basis, since the server is aware of the average request interarrival time for each video. Since we model the request arrivals as a Poisson process, which is defined by a single variable that is historically called λ, we call this variation "λ Patching". In the following section, the optimization by recursive application of the patching concept is introduced as a means for achieving additional server load reduction. We accept that some near video-on-demand-like traffic is generated with additional patch streams to achieve these additional gains. We call this optimization option "Multilevel Patching".

Section 3.4 addresses two questions:

- whether the redundancy of cache-based distribution systems is too cost-intensive, and
- whether a combination of patching and caching is worth-while

It presents a favorable argument for the application of patching in a distribution hierarchy. A simple example distribution hierarchy is considered, which can be analyzed with complete knowledge of server and networking costs. When these costs are considered, a strong indica-

tion is achieved that patching and caching are worth-while models for a distribution system. The chapter is concluded with a discussion of the implications of the presented techniques for decentralized distribution systems that make use of caches.

3.1 Refined Description of Patching

For the exploitation of multicast in TVoD systems, Hua et.al. invented *patching*. The basic approach, presented in [HCS98], is the creation of a multicast group for the delivery of a video stream to some requesting clients.

Subsequent requests in a temporal interval after each multicasted movie are not served by transmitting the same movie again. Instead, the client is provided with sufficient information to join the initial stream, and an additional patch stream for the missing initial portion of the movie. These subsequent clients provided with patch streams use local cyclic buffers to delay play-out of received main multicast portion of the movie. If another client requests the same video shortly after the start of this transmission, this client starts storing the multicast transmission in a local cache immediately. The server sends a unicast stream to this client containing the missing initial portion of the video, until the cached portion is reached. Then, the client uses its cache as a cyclic buffer.

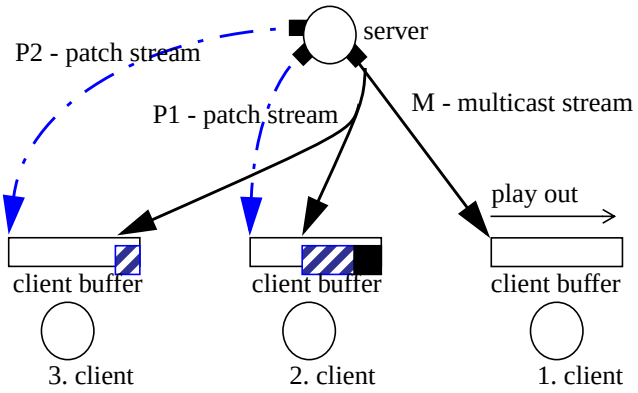


Figure 10: Buffer Usage in patching

Figure 10 tries to demonstrate the effects of patching at the client; in contrast to typical ordering, the sequence of clients that request a movie from the server is ordered right-to-left, in order to maintain the a left-to-right representation for buffer consumption (“play-out”). In the figure, the 1. client arrives first and receives the stream M, which carries the movie sequentially, starting with the first byte. The white client buffer indicates that it remains completely unused in this case. The 2. client arrives a bit later, joins the transmission of stream M to the 1. client (to receive the later parts of the video), and receives an additional unicast patch stream P1 for the missing initial portion. The striped box indicates the amount of buffer space that is required to store those parts of the movie that are received from stream M. As soon as the initial parts have been shown and P1 ends, the remaining part of the movie is shown from the buffer, which will then act as a cyclic buffer for the remaining portion of the stream. That length of the remaining portion is indicated by the black box. The arrival of the 3. client is handled in the same way as the 2. client, except for the difference that the patch stream P2 delivers

a larger part of the movie to the client than stream M. Because of this, there is no possibility for using the buffer as a cyclic buffer.

In the example above, it is assumed that a patch stream may be as long as a complete movie. In a realistic scenario, there is a maximum time after a full movie start, after which the next request is answered by repeating the complete movie multicast stream, instead of sending more patches in parallel. [HCS98] presented two approaches for this, called *greedy patching* and *grace patching*. Greedy would buffer as much of the original movie as possible, allowing for patch stream length up to the complete length of the movie. Grace is an adaptable approach that assumes a maximum buffer size at the client.

3.2 λ-Patching

The first Patching presentation gave the impression that enlarging buffers would always reduce the required server bandwidth. However, the number of patch streams and thus, the number of potential concurrent streams, increases with a movie's popularity, leading to an increased number of concurrent patch streams after a full stream has been started. Server load becomes bursty. This implied to us that an optimal restart time must be related to the movie popularity.

To get an intuitive understanding of this thought, imagine a somewhat popular movie title[1] that is served to clients using patching. The size of the connected user community and the title's popular result in an average request interarrival time that is very small compared to the movie title's length. When a title is requested and the server decides to serve it as a full stream, a multicast stream is initiated immediately. For several subsequent requests, patches to this stream are delivered in addition to the multicast stream. But whenever a request for the title arrives at the server, it is still serving approximately half of the patch streams for the title. Also, since patch streams can run as long as the size of a patching window, the server may supports some patches belonging to the previously started full stream which terminate inside the stream's patching window. Visually, while new requests are served as patches for a full stream, the average number of patches belonging to this full stream is growing linearly, when no more requests are accepted. the number of falling linearly with the same slope. Since patching belonging to the previous full stream are leaving the system at the same rate that patching belonging to the new full stream are started, the average number of patch streams is constant and depends on the patching window size and the request rate. Obviously, if the patching window size is shrunk, the average number of patch stream in the system is reduced, reducing the server load as well. However, full streams must be started more frequently if the window is shrunk, and the number of concurrently active full streams rises, increasing the server load. This implies that the patching window size can be optimized depending on the life time of a full stream and on the request frequency for a title.

Figure 11 illustrates the problem. The X-axis of the Figure 11 indicates the time that passes for the server. The Y-axis of the figure is the position in the movie M, from position 0 to the end position. This position is expressed in time. X- and Y-axes use the same scale in our figures; thus the distance between the movie start and the position of a frame in a movie on the Y-

1. a very popular movie title is better served by a batching approach, and an unpopular title is better served by unicast

axis is identical to the distance between the start time of a movie and the display time of this frame on the X-axis, under the condition that the movie is streamed at normal display speed. In the figure, the solid diagonal lines denote the progress of stream transmission for complete movies, the dotted diagonal lines denote the progress of patch streams. The vertical, numbered vertical lines sample the number of concurrent streams at a given time. The fact that the lowest number of concurrent streams (4) is neither achieved with the largest nor with the smallest overlap of complete movie transmissions is meant to demonstrate the starting point of our optimizations: the number of concurrent multicast and unicast streams has a non-trivial minimal value.

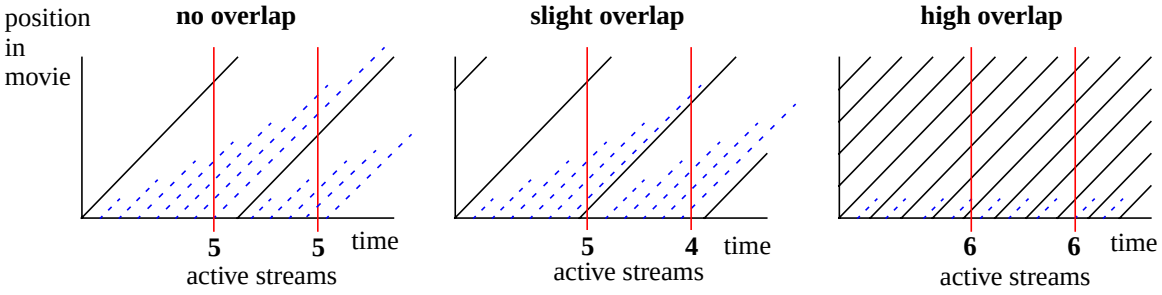


Figure 11: Hints that patching window size may have an optimum

Our investigations, started not realizing the relationship with the skyscraper techniques ([VI96], [AWY96], [HS97], [EV98], [JT97], [PCL98a], [PCL98b]) were first aimed at an understanding of the traffic generated by movie delivery via Patching. Then, we have analyzed the optimal server load based on the knowledge of request frequencies.

One of the important pre-conditions for this investigation is the assumption that the interarrival times of the user requests are Poisson-distributed. This is not necessarily correct. However, we argue that this simplification can be made because of the time intervals that are affected by this abstraction.

First of all, we assume that the server is the instance that decides when it is most efficient to send a complete movie rather than a patch stream to a client. We call the time between two consecutive movies the patching window. The size of this window varies with respect to movies popularities, and a correctly or incorrectly chosen patching window size affects the server's subsequent actions only until the movie transmission ends for the movie that is transmitted following the patching window. The computation of the following window's size can be made independently from the previous. This means that a server must take only a few preceding hours into account for a window size decision.

We believe (but can not prove) that the user behavior in a true video-on-demand system on this time scale appears random to an observing server and that movie's popularities are changing only gradually on this scale. On the other hand, we believe that there is change throughout the day, which implies that information about previous user hits preferences should not be taken into account by the server for more than a few window sizes (resp. hours). We ignore this issue in computations on the basis that the server's decisions that we propose can be made whenever a request for a video arrives, based on knowledge that has sufficient short-term validity, e.g. for a few hours.

Given these conditions, it seems appropriate to keep only information about the interarrival times for requests to each individual movie titles, and to make no assumptions about relations between subsequent requests. To model such a condition, the Poisson distribution is appropriate.

F	length of movie	sec
Δ_M	time interval between multicast starts (also called patching window size)	sec
$\Delta_U = 1/\lambda$	expected time interval between video demands (unicast starts), following the negative exponential distribution	sec
B	buffer length at the client	sec
C_U	cost of unicast stream at server	EUR/sec
C_M	cost of multicast stream at server	EUR/sec
S_U	unicast stream setup cost at server	EUR
S_M	multicast stream setup cost at server	EUR

Table 1: Terms and definitions of the calculations

For our calculations, we call the average interarrival time of the Poisson-distributed arrivals for a given movie title $\Delta_U = 1/\lambda$; this time is also our value for the current popularity of the video (i.e. average interarrival times are observed by the server and assumed to be Poisson). Other symbols that are used in the following calculations can be found in Table 1.

We simplify the patching model by starting multicast streams in cycles of length Δ_M rather than on-demand. This implies a near video-on-demand (NVoD) transmission model for the multicast transmissions as in the skyscraper papers. It provides several convenient simplifications to computations, e.g. that the expected value for the number of concurrent streams is time-independent and that the end times of patch streams are Poisson distributed.

The simplification has the negative effect that the resulting calculations over-estimate the necessary number of concurrent streams. In patching, a complete movie transmission is started only when the next user request arrives after a patching window; our model requires an immediate transmission and in this way, requires a lengthening of each patch stream by an average interarrival time. This over-estimation has an increasing effect with a growing interarrival time (i.e. with a decreasing popularity).

In the following, we want to optimize server load in terms of numbers of concurrently active streams under the given conditions.

3.2.1 Expected Patch Stream Length

We start with the computation of the patch stream length because the expected value of the number of unicast streams that are started in each interval of length Δ_M between two multicast stream starts is Δ_M/Δ_U. In all subsequent considerations, we assume that Δ_M/Δ_U is large. If this is not the case, the simplification of using fixed restart times for the computations is misleading when in comparison to an implementation of patching without this simplification.

Assuming that one full multicast stream starts at time 0, the length of each unicast transmission can be calculated as follows:

$\forall t \in [n\Delta_M, (n+1)\Delta_M)$: $\quad length(t) = t \bmod \Delta_M$

because as shown in Figure 12, the length is identical to the length of the missing part of a movie. We compute the expected value of the patch stream length and find that it is $(1/2)\Delta_M$.

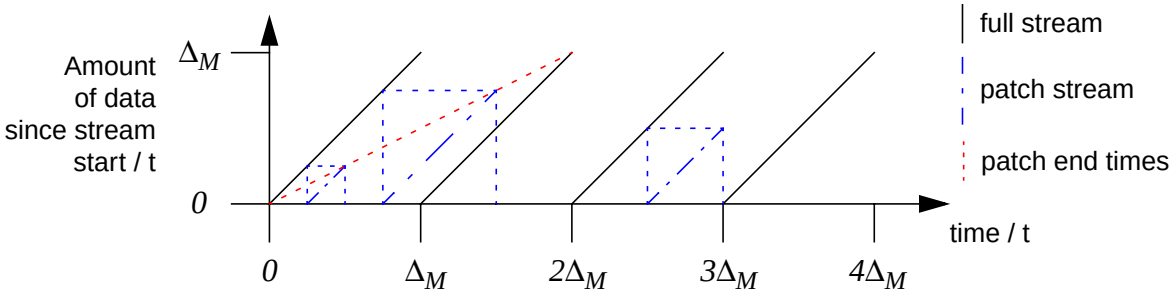


Figure 12: Patching with Cyclic Restart

3.2.2 Expected Number of Active Patch Streams

The expected interarrival time of streams is Δ_U. Examining the indicated line of patch stream end times in Figure 13, intuition demands that the average number of streams which are concurrently active is $\Delta_M/(2\Delta_U)$. The expected value of the number of streams that are concurrently active at a given time t is less intuitive (although the result is the same).

We examine the interval of possible starting times for streams that can still be active at the given time t.

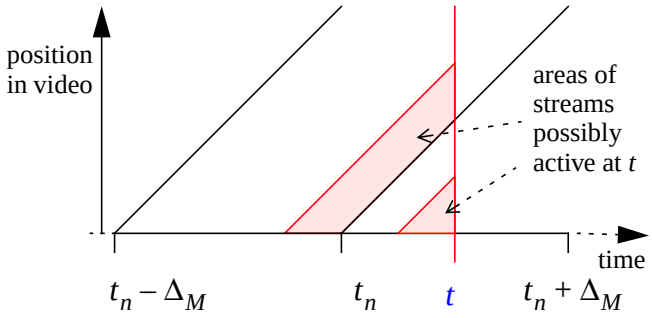


Figure 13: Expected start time intervals for active streams at time t

This interval is defined by two sub-intervals. One includes the streams that are started in the same interval $[t_n, t_n + \Delta_M)$ where t_n is that latest multicast stream starting time before t and still active at time t. The other includes the streams that have been started in the interval $[t_n - \Delta_M, t_n)$ and that are still active at time t. In our model, no patch streams from earlier patching windows can be active at time t because we assume a constant patching window length, and patch stream can never run longer than the length of one patch stream window. With earlier definitions, this provides the following time ranges from which starting points U_t: can still be relevant at time t

These intervals are always disjoint, and their combined length is $|U_t| = 1/2 \cdot \Delta_M$

$$U_t = \left[\frac{t + t_n - \Delta_M}{2}, t_n\right) \cup \left[\frac{t + t_n}{2}, t\right)$$

Since the Poisson distribution defines that the expected number of arrivals in any interval T is T/Δ_U, this provides the expected number of active streams at time t, i.e., the number of streams that are started in U_t, which is $|U_t|/\Delta_U$. This results in equation (1), calculating the expected number of unicast streams active for any t,

$$N_u(t) = \frac{\Delta_M/2}{\Delta_U} = \frac{\Delta_M \lambda}{2} \tag{1}$$

equal to the average number of concurrent unicast streams.

3.2.3 Optimizing Δ_M

Since all complete multicast streams have length F, $N_m(t) = F/\Delta_M$ multicast streams are concurrently active at each time. Together with equation (1), we have the overall number of concurrent streams,

$$N(t) = N_m(t) + N_u(t) = \frac{F}{\Delta_M} + \frac{\Delta_M}{2\Delta_U} \tag{2}$$

By adding server stream maintenance costs and server stream setup costs for multicast and unicast streams, we get

$$\text{Cost}_{\lambda\text{-patching}} = \frac{S_M}{\Delta_M} + \frac{S_U}{\Delta_U} + C_M \cdot \frac{F}{\Delta_M} + C_U \cdot \frac{\Delta_M}{2\Delta_U} \tag{3}$$

the overall server streaming cost. We can now use the expected cost by computing an optimal value for Δ_M. It depends on the current popularity of the video, which is expressed by $\Delta_U = 1/\lambda$. We get

$$\begin{aligned} 0 = \frac{\delta}{\delta\Delta_M}(\text{Cost}_{\lambda\text{-patching}}) &= -\frac{S_M + C_M F}{\Delta_M^2} + \frac{C_U}{2\Delta_U} \\ \Leftrightarrow \Delta_M &= \sqrt{2 \cdot \frac{S_M + C_M F}{C_U} \cdot \Delta_U} \end{aligned} \tag{4}$$

By neglecting setup costs and assuming $C_M = C_U$, this can be simplified for an approximation of the optimal value of the client buffer's size as a time B_λ. It depends on the popularity and on the length of a video :

$$B_\lambda = \Delta_M = \sqrt{2 \cdot F \cdot \Delta_U} = \sqrt{\frac{2F}{\lambda}} \tag{5}$$

(under the condition that the client can receive 2 concurrent streams).

We derive Δ_M directly from given figures, so that a video server can recalculate Δ_M for every given film or change in request rate or even bandwidth costs. This approach is more easily applied in the real-world than in simulations.

Example

To demonstrate the use of these equations, consider the following example: let multicast and unicast streaming costs be equal, multicast stream setup costs be $C_M \cdot 0.5\text{sec}$ (i.e., setup is worth half a second of streaming) and unicast stream setup costs be $C_U \cdot 5\text{sec}$. Let the film be a popular movie of 4200 seconds with an average request interarrival time Δ_U of 3 sec. This results with equation (4) in an optimal temporal distance Δ_M between multicast restarts of about 159 seconds (equation (5) calculates the same). The server streaming cost for this Δ_M is equivalent to about 53.11 concurrent streams (equation (3)), with multicast streams cost equivalent to 26.3 concurrent unicast streams, including multicast setup costs.

3.2.4 Given Limits

There is obviously a lower limit to the frequency with which streams need to be started even under very high loads: since there is a limit to the user perception of lag in stream acquisition, it is acceptable to delay the stream start for a few seconds without giving the user the impression of an NVoD system. This imposes a lower limit to reasonable Δ_U values that we did not exploit in our calculations.

As every client eventually has to buffer Δ_M of video, a centralized VoD system must require a minimum client buffer size from the end systems. This minimum client buffer size is an upper bound to Δ_M. This restriction does not apply in our target system which is based on caches without centralized control that store only complete movies. In conjunction with cache servers such buffering limits are not relevant when a caching strategy is applied that stores all the requested movies unconditionally. In conjunction with caching strategies that cache movies only under certain conditions, the problem applies also to cache-based distribution system, since the cache servers must maintain a amount of buffer space Δ_M for each title that is not cached. Alternatively, the clients could be required to do so, but this approach will not be considered here.

3.3 Multilevel Patching

In this section we extend the patching algorithm by additional multicast patch streams. This extension of patching is called Multilevel Patching. It is demonstrated that the server load can be traded for client network bandwidth.

A variety of schemes that are more difficult to evaluate can be imagined. We could also do harmonic or binary approaches to the segmentation of streams into patches, or further optimization of non-consecutive portions of streams. All of these approaches tend to work primarily for broadcast-like distribution, and waste bandwidth when they are applied to true video-on-demand for infrequently requested titles without major changes.

Among the approaches that could be adapted are the stream merging approaches. These allow transmission of movies at various speeds. Since our display quality is considered fixed, this means that a stream slowdown is not possible, but that a speed increase is possible. It implies also that the burstiness of the bandwidth at the server will increase. There is a potential for savings if patch streams can be joined with neighbor patch streams by increasing their

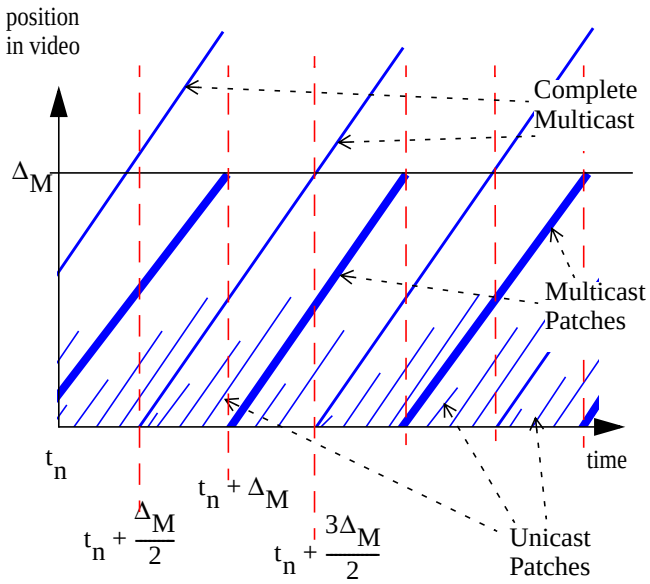


Figure 14: Stream setup example with first multicast patch

speed. In a real-world implementation this seems rather complex; we have not evaluated this option.

3.3.1 First Multicast Patch Stream

We assume that a client is able to receive up to three streams in parallel. Then, we extend the patching algorithm for the server by the rule: "in every interval $T_n = [t_n, t_n + \Delta_M/2)$ between the starts of two complete multicast streams multicast an additional patch stream at $t_n + \Delta_M/2$, and play it for a length of Δ_M".

The extension requires the client to listen to a complete multicast stream, potentially one unicast patch and potentially one additional multicast patch. This increases peak receiving load on the client up to three concurrent streams, demanding for higher bandwidth between client and server and higher client computation power. The buffer requirements do not change, as the received amount of data to be buffered is still a maximum Δ_M, although eventually written concurrently in two portions.

Chosen Position of First Multicast Patch

Unicast patches deliver only the amount of data not available from the last multicast stream (including complete multicast streams and multicast patch streams). Their average length and with that the average number of concurrent unicast streams is proportional to the gap between multicast streams. We therefore start a multicast patch in the middle of two multicast stream starts to decrease the average required length of unicast patches.

With a multicast patch halfway in between two complete streams, unicast patches only patch a maximum gap of $\Delta_M/2$. In the same way as seen above, this gives us an expected number of $(\Delta_M/2)\lambda/2 = \Delta_M/4\Delta_U$. The average number of concurrent unicast streams over an arbitrary interval with one multicast patch is halved.

Chosen Length of First Multicast Patch

There are two cases, depending on the position of the client's request time in the interval between two complete multicast streams.

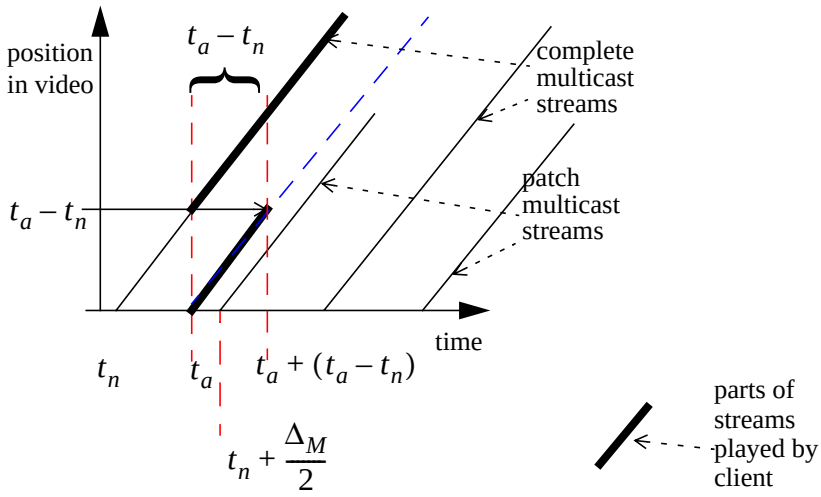


Figure 15: Request at time $t_a \in \left[t_n, t_n + \frac{\Delta_M}{2}\right)$

- If the client requests a video at a time t_a in the first half of an interval between two complete multicast streams (Figure 15), it listens to the unicast patch stream and to the complete multicast stream, immediately playing the unicast. The multicast stream is buffered and played with a delay of $t_a - t_n$.
  These clients do not use the multicast patches the server provides.
- If the client requests a video at time t_b in the second half of an interval between two complete multicast streams (Figure 16), it listens to the unicast patch stream, to the last multicast patch stream and to the last complete multicast stream. It immediately plays the unicast

stream, the two multicast streams are buffered and played with a delay of $t_b - (t_n + \Delta_M/2)$ for the multicast patch $t_b - t_n$, respectively, for the complete multicast.

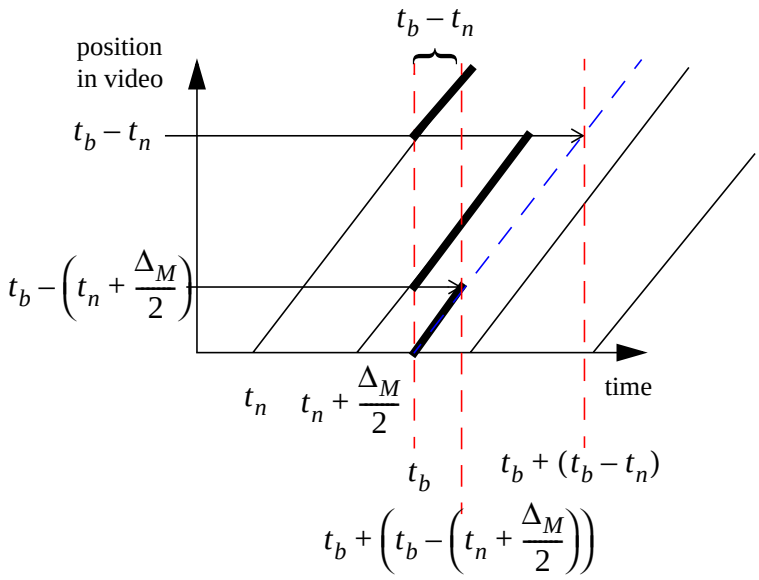


Figure 16: Request at time $t_b \in \left[t_n + \frac{\Delta_M}{2}, t_n + \Delta_M\right)$:

Figure 6 shows that the multicast patch at $t_n + \Delta_M/2$ eventually has to patch the video data of the interval $[t_b - (t_n + \Delta_M/2), t_b - t_n)$ with $t_b \in [t_n + \Delta_M/2, t_n + \Delta_M)$, which gives that the latest video data possibly to be patched are at $t_n + \Delta_M - t_n = \Delta_M$.

Thus, the multicast patch has to cover an interval of data to be patched of $[0, \Delta_M)$, being twice as long as a unicast patch starting at the same time would have to be.

Evaluation of First Multicast Patch

With a fixed client buffer, but with $3/2$ of peak receiving load compared to original patching, we introduced multilevel patching with one intermediate multicast patch. With the halved unicast load and with one additional multicast patch of length Δ_M starting every $\Delta_M/2$, the required bandwidth cost at the server is $C_M \cdot \frac{F}{\Delta_M} + C_M \cdot 1 + C_U \cdot \frac{\Delta_M}{4\Delta_U}$

The gain over non-multilevel patching on the server is as below.

$$C_U \cdot \frac{\Delta_M}{4\Delta_U} - C_M$$

This will be a positive value for large Δ_M/Δ_U. In our example, we get 27.4 multicast streams and 13.25 unicast streams concurrently on the server.

Including the stream setup costs for multicast and unicast streams at the server, the cost for multilevel patching is :

(6) $$\text{Cost}_{\text{1st mc-patch}} = \frac{2S_M}{\Delta_M} + \frac{S_U}{\Delta_U} + C_M \cdot \frac{F}{\Delta_M} + C_M + C_U \cdot \frac{\Delta_M}{4\Delta_U}$$

With equation (3), this is a gain of :

$$\text{Cost}_{\lambda\text{-patching}} - \text{Cost}_{\text{1st mc-patch}} = -\frac{S_M}{\Delta_M} - C_M + C_U \cdot \frac{\Delta_M}{4\Delta_U} \quad (7)$$

This again will be a positive value for large Δ_M / Δ_U.

For our example above, equation (6) gets server costs for λ patching with a first multicast patch as an equivalent to 40.89 concurrent streams, saving in this example an equivalent of more than 12 streams from non-multicast patching.

3.3.2 n-th Multicast Patch

To introduce the first multicast patch for multilevel patching, we had to extend the available maximum client bandwidth to $2 + 1 = 3$ streams, which has to be fully available during a short time immediately after requests. But if clients can receive $W > 3$ concurrent streams, we can introduce $W - 2$ multicast patch streams by applying the multicast patch recursively. The resulting characteristics of multilevel patching with n multicast patches are:

- peak receiving load: $W = n + 2$
- a time interval of $\Delta_M / 2$ between multicasts, resulting in an average number of concurrent unicast streams on the server of

$$\frac{\Delta_M}{(2^{n+1}\Delta_U)}$$

- Server bandwidth cost of

$$C_M^S \cdot \frac{F}{\Delta_M} + nC_M + C_U \cdot \frac{\Delta_M}{2^{n+1}\Delta_U}$$

- Server bandwidth and stream setup cost of

$$\text{Cost}_{\text{nth mc-patch}} = \frac{(n+1)S_M}{\Delta_M} + \frac{S_U}{\Delta_U} + C_M \cdot \frac{F}{\Delta_M} + nC_M + C_U \cdot \frac{\Delta_M}{2^{n+1}\Delta_U}$$

- With a gain over non-multicast patching of

$$\text{Cost}_{\lambda\text{-patching}} - \text{Cost}_{\text{nth mc-patch}} = -\frac{S_M}{\Delta_M} + C_U \cdot \left(1 - \frac{1}{2^n}\right) \cdot \frac{\Delta_M}{\Delta_U} - n \cdot C_M$$

Again, these formulae are valid only for large Δ_M / Δ_U. Also, saved unicast bandwidth soon will be outweighed by additional expenses in multicast path tree setup and bandwidth. But if we consider the equations, we get a theoretical optimum of savings over non-multicast patch-

ing

$$0 = \frac{\delta}{\delta n}(C_U \cdot \left(1 - \frac{1}{2^n}\right) \cdot \frac{\Delta_M}{\Delta_U} - n \cdot C_M) \tag{8}$$
$$\Leftrightarrow n = log_2\left(\frac{C_U \cdot \Delta_M}{\Delta_U(S_M/\Delta_M + C_M)}\right) - 1$$

The optimum for n here is computed for a fixed Δ_M, as for now we do not optimize the two-dimensional tupel (Δ_M, n).

The multilevel patching scheme could easily be extended to chose n according to a client's buffer and available bandwidth, as existing streaming approaches like MPEG-4 ([ISO98]) support dynamic setup for multi-stream connections. This would allow for a scheme to individually set up multilevel-patching for each client, dynamically calculating the appropriate length of patches.

Example

For our example movie above, equation (8) gives an advice to use the fourth (or fifth) multicast patch:

$$n = log_2\left(\frac{C_U \cdot 159}{3\left(\frac{5C_U}{159} + C_U\right)}\right) - 1 = log_2\frac{159 \cdot 159}{3 \cdot 164} - 1 \approx 4.7$$

This would result in a multicast patch every 9.9 seconds (resp. 5 seconds). Using the fourth (fifth) multicast patch on our example, we get server streaming and stream setup costs equivalent to 32.4 (32.6) concurrent streams, which means further savings of 8.4 concurrent streams over first multicast patching. The video server with n-th multilevel patching in this theoretical example could provide TVoD while being only about ten streams more expensive than NVoD at a granularity of 159 seconds (26.4 concurrent multicast streams). As stated above, this is in trade-off to the expense of 159 seconds buffer and the triple (resp. $7/2$) required burst bandwidth on every client.

The results of Section 3.2 and Section 3.3 can be used for single server systems as well as for our distribution system that is based on caches. However, requirements for multicast communication between servers and end-users differ from the requirements on inter-cache communication. In the first case, a real-time transmission of data is necessary which should not suffer from excessive packet loss, but some packet loss is permissible. In the second case, on the contrary, packet loss in not acceptable when a video is transferred to the cache.

3.4 Motivation of Gleaning for Caching Hierarchies

Caching is a motif of this thesis. Considering the vast number of broadcasting approaches that are listed in the introduction of this chapter, this motif needs a foundation. It is hardly questionable whether a cache-based architecture provides stability to a distribution system. Assuming that an appropriate replacement strategy is applied, the caches should store the most popular video titles and thus, be able to serve the requests of a majority of users even when their uplink

to the origin of the data, or the original server itself is broken. Similarly, most users are not affected by downtime of a cache elsewhere in the system, and even the customers that usually connect directly to that cache may still be able to receive movies, e.g. from the original server.

It is not questionable either whether a decentralized cache-based architecture can provide a more dynamical potential for expansion to a VoD infrastructure. Rather than putting the complete infrastructure in place with future expansions in mind, caches may be owned by independent service providers that compete for the service to a user community. Just like video rental shops that are organized in chains or privately owned, cache servers can be installed locally when the business potential rises; it is also easier to move or discard cache servers of limited size when the business potential decreases in one location.

It is questionable whether caches waste resources by replicating content into several cache servers, which introduces potentially a large amount of redundancy. Stated like this, the question originates in the cost of storage and perhaps in the “abundant bandwidth” assumption. It does not take the cost of networking into account. This section presents our combined patching and caching approach called *gleaning* and a view of a distribution system that considers the cost for storage and for networking infrastructure.

3.4.1 Design of Gleaning

Gleaning has two goals:

- to form the basis for the technical solution to bringing the advantages of patching to distribution systems that rely on existing clients that are not patching-capable or can not waste large amounts of their memory on movie buffering
- to combine the availability gains and bandwidth-saving features of caching with the efficient bandwidth usage of patching

The combination can be made without much consideration to overlapping functionality. Gleaning is a distribution mechanism that reduces the load of the sending party, while caching strategies are meant mechanism to manage the storage space of the cache server. The bandwidth-reducing effect of both on the link that connects sending server and receing cache is not competing either if multicast techniques like the Internet’s IP multicast is used, which is based on explicit joining of the receiver, and not on sender-initiated trees.

Gleaning works as follows: Cache servers are deployed as proxy caches, i.e. client will always connect to their proxy server to access data on the origin server. If it is intended to deploy the caches in a user-transparent manner, this can be achieved without built-in proxy functionarlity as well, by enabling the origin server to learn about the cache servers and the approximate distances in the network. Based on this information, the origin server can re-direct commands of the client to the proxy cache.

In the distribution system that is built in this way, each cache server has a dedicted parent node, either the origin server or another cache server, thus forming a strictly hierarchical system.

When a client requests a movie title from its proxy cache, and it is stored partially or completely in the cache, it is delivered to this client immediately, unless the caching strategy has marked the title for removal or some other problem is experienced with the copy (e.g. an

exhausted disk bandwidth). Since we assume that gleaning works with complete movie titles only, the presence of a partial title implies that the rest of the movie is in transfer from an upstream cache or form the origin server.

When a client requests a movie title from its proxy cache and it is not present, the proxy cache tries to initiate the transfer from the next upstream cache or the origin server, respectively. To prevent an exhaustion of the uplink by this forwarding of requests, the caching strategy can decide to forbid the request forwarding; the client's request will be refused in this case. If the request is forwarding to a potential sender, and the title is not currently delivered to any cache or end-system, and if sufficient bandwidth is available at the sender and in the network, a new multicast stream is iniated, and the client is invited to the multicast session. If the cache server decides to keep the title, it joins the multicast stream as well

If the stream is already being delivered to a cache server or client, and the sender decides that the patching window for this stream is still open, it orders the cache server to join that multicast stream. Additionally, it starts the transmission of a patch stream to the proxy cache. The proxy cache has to set aside sufficient buffer space for the cyclic buffer to hold the length of the patch stream, even if it does not cache the movie; the stream is delivered as a unicast stream to the client.

Designed in such a way, a cache server in a gleaning system will not be overloaded more easily than in a typical cache-based distribution system. In the worst case scenario that the cache can never join a multicast, it will behave at least as good a cache server in a unicast cache-based distribution approach. Similarly, a sending server will never experience more hits than in a pure patching approach. In the worst case, each proxy cache forwards the requests of a single client that retrieves unpopular movie titles.

3.4.2 Cost comparison

We examine the feasibility of patching and caching by modeling analytically the necessary effort in an example hierarchical movie distribution scenario. First, we calculate cost functions for various approaches of serving movies to users in hierarchical distribution systems with the topology of binary trees. Then, we apply this analysis to an example system with somehow realistic features.

Figure 17 is a sketch of the base model topology central server *CS*, optional cache servers N_i^t with an index *i* at depth *t* in the binary tree, and network links E_i^t. Table 2 lists the symbols that are used in the formulas, and Table 3 presents the formulas for calculating the cost of the distribution systems. The most important limitations of the model are summarized below, but still, this analysis motivates us to realize our approach called *gleaning*, which integrates caching with patching. Basically, cache servers in *gleaning* behave in the same way as cache servers in a straight-forward cache-based distribution system, and it is possible to apply identical removal decisions within their caching strategies. The transfer of data into these caches differ with *gleaning*, by making use of multicast distribution, and of the ability of cache servers to hide reordering operations within a stream from the end-systems. As described with *patching*, the original sender of the movie decides the order of delivering chunks of the movie, but by maintaining cycling buffers for the later parts of the movies, the receiver is able to view the title in

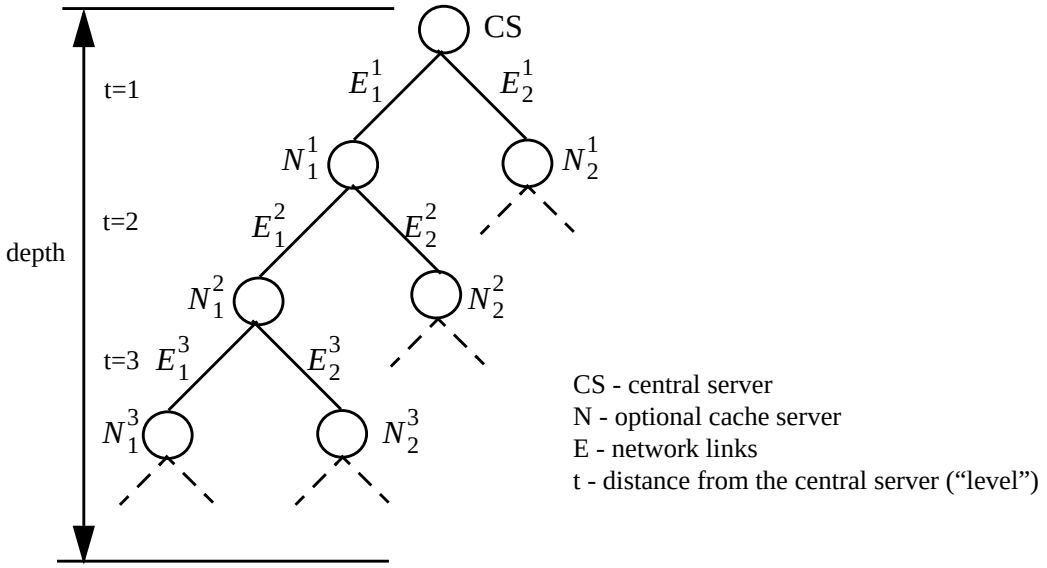


Figure 17: Binary tree model for the optimization of total networking and storage cost

order. In *gleaning*, the cache server acts as a patching client, and sequentializes the play-out of the movie to the end-system. We get a strong hint to combine caching with *patching* in the example below, for a VoD system with rather realistic characteristics, following the assumptions of the analysis.

The effort to set up the system is modeled as an abstract “cost” for basic server installations (including central server and cache servers), cost of server support for concurrent stream deliveries, the cost of concurrent streams support by each network link, and cost for the storage of movies in cache servers. As we assume all movie files to be optimally located in the caching hierarchy, there is no cost for transporting the movies to store and cache and for unnecessary copies. There are several noteworthy aspects to this assumption:

- assuming a perfect distribution of movies to cache servers according to their long-term relevance would also render movements due to relocation minimal
- for a downstream movement, caches that work according to our approach do not generate additional network load because they work in write-through mode - upstream movement is certainly missing
- if caching strategies are not sufficiently elaborate (or centrally controlled), they will react to short-term or at least to day-time variations in the request patterns, these calculations will be extremely optimistic

The numerical optimization assumes a distribution of movie hit probabilities according to the Zipf distribution. Although various papers state that the Zipf distribution describes the distribution of hit probabilities at any given time very well, a caching architecture is unable to achieve a distribution according to Zipf.

- The relevance of movies is changing with respect to other movies, which implies that their index value in the Zipf distribution is changing,
- Hit rates do not typically conform perfectly to the Zipf distribution because of user behavior. The divergence is greater for small user populations, which means that distribution sys-

tems without an exchange of hit rate information will estimate a movie's popularity less exact than a centrally coordinated system.

- Movies must be relocated between cache servers according to their estimated relevance. This may be done predictively (which reduced accurateness of the estimation), so the optimal location for each movie is achieved timely, but such relocations do still incur additional network and server load.
- Homogenous distribution systems are unrealistic.
- Not all movies have equal length and data rate.

Note, that a non-hierarchical approach will probably result in additional savings but for hierarchies, any algorithm should be unable to reach the optimum that can be computed numerically from the formulas in Table 3.

To verify the effects of these computations, we present an example that demonstrates the vast options for savings. This example is simplified from the reality that we envision with the combination of *patching* and caching. For example, we assume that *patching* is implemented in the clients, which is not realistic in a widely distributed network of heterogeneous clients, since technical advancement in CPU power and storage space will not lead to an increased capacity of the low-end devices, but rather to the creation of more compact devices of a similar computing power and storage space..

symbol used	meaning in the formulas	symbol used	meaning in the formulas
S_0	Basic cost of a server/cache server installation.	S_1	Cost for one supported stream of a server.
C_t^E	Cost for one supported stream on a network link at level *t*.	C_t^N	Cost for the storage needed to store one movie in a cache server.
M	Number of available movies.	$P(m)$	Hit probability of movie *m*.
t(m)	Optimal tree level for caching movie *m*.	r(m)	Optimal patching window for movie *m*.

Table 2: Elements used in formulas

distribution method	calculated cost formula
unicast directly from central server	$S_0 + 2^d \cdot S_1 + 2^d \sum_{t=1}^{d} C_t^E$

Table 3: Analysis of cost effects of patching on caching hierarchies, cf. "Analytical Distribution Model - Binary Tree" on page 153

distribution method	calculated cost formula
unicast with caches	$\left[\sum_{t=1}^{d-1}\left(2^t \cdot \bigcup_{m \in M} \delta_{t(m)}(t)\right)\right] \cdot S_0 + 2^d \cdot S_1 + \sum_{m \in M}\left[P(m)2^d \sum_{t=t(m)+1}^{d} C_t^E + 2^{t(m)} C_{t(m)}^N\right]$
greedy patching from central server	$S_0 + \left[2^{d-1} + \sum_{m \in M}(1-(1-\eta_m)^{2^d})\right] \cdot S_1 + \sum_{m \in M}\sum_{t=1}^{d}\left[\left(2^t - 2^t \cdot (1-\eta_m)^{2^{d-t}} + 2^{d-1} \cdot \eta_m\right) \cdot C_t^E\right]$,where $\eta_m = P(m)$
patching with limited buffer from central server	$S_0 + \left[\sum_{m \in M}(2^{d-1} \cdot \eta_m + r(m)(1-(1-\eta_m)^{2^d}))\right] \cdot S_1 + \sum_{m \in M}\sum_{t=1}^{d}[\eta_m \cdot 2^{d-1} + (2^t \cdot r(m) \cdot (1-(1-\eta_m)^{2^{d-t}})) \cdot C_t^E]$,where $\eta_m = \frac{1}{r(m)} \cdot P(m)$
gleaning	$\left[\sum_{t=0}^{d-1}\left(2^t \cdot \delta(\bigcup_{m \in M}(t = t(m)))\right)\right] \cdot S_0 + \sum_{m \in M}(2^{t(m)} C_{t(m)}^N) + \left[\sum_{m \in M}(2^{d-1} \cdot \eta_m + 2^{t(m)} \cdot r(m)(1-(1-\eta_m)^{2^{d-t(m)}}))\right] \cdot S_1 + \sum_{m \in M}\sum_{k=1}^{d-t(m)}[(\eta_m \cdot 2^{d-1} + 2^{t(m)} \cdot 2^k \cdot r(m) \cdot (1-(1-\eta_m)^{2^{d-t(m)-k}})) \cdot C_k^E]$,where $\eta_m = \frac{1}{r(m)} P(m_m)$

Table 3: Analysis of cost effects of patching on caching hierarchies, cf. “Analytical Distribution Model - Binary Tree” on page 153

In our example, the movie probabilities are distributed according to the Zipf distribution:

$$P(drawm_m) = z(m) = \frac{C}{m}, C = \sum_{m \in M} \frac{1}{index(m)}$$

Besides the predefinitions from the analytical model, we choose values for the individual parameters. These values are chosen rather speculatively; our orientation were the product prices that we paid for our department’s commercial video server.

- 500 different movies
- 2^{20} active users (i.e. a binary distribution depth of 10, where most nodes do not contain a server)
- a cost of 25000 $ for a basic server installation
- a cost of 100 $ for each concurrent high quality movie stream supported by a server
- a cost of 350 $ for each concurrent high quality movie stream supported on a network link
- a cost of 1000 $ for storage to hold one high quality movie

The location of the caches in the distribution hierarchy for examples 2,”unicast with caches”, and 5, “gleaning”, was not optimized. Rather, the caches were moved heuristically upstream until no immediate gain was perceived any more. For the example 2, “unicast with caches”, the

approach “installed” caches at levels *t*=12, 10, 8, 6 and 4 of the binary distribution tree model in the order of decreasing movie popularity. For the example 5, “gleaning”, the approach “installed” caches at levels *t*=9, 7, 3, 5 and 1. The heuristic prohibited to choose the level 0 for the least popular movies which would have been roughly three quarters of all movies.

Modeled Distribution Method	Calculated System Cost
unicast from central server	7,445 Mio $
unicast with caches	4,664 Mio $
greedy patching from central server	3,722 Mio $
patching with limited buffer from central server	375 Mio $
gleaning	276 Mio $

Table 4: Example for theoretical effect of the various methods

These numbers indicate, that there are scenarios with a large potential for savings in the *gleaning* technique. When (costly) caches are introduced in a *gleaning* distribution system, savings are made with much less expensive necessary system links and storage space (cf. the last two rows in Table 4).

Although this model and these numbers are quite illusionary, and we can not expect clients that implement *patching* buffers and *patching*-capable protocols, this potential for savings demonstrates that:

1. the use of cache servers generates savings that make up for their installation cost
2. *patching* with optimized window sizes is the major advancement in savings
3. The most important issue for our architecture is:

The installation of caches in conjunction with *patching* does not eliminate the effect of *patching*. With an appropriately dimensioned cache server, it will even increase the savings by keeping the most popular titles in the cache. Thus, we can proceed to build a wide-area caching architecture that relies on *patching* for wide-area distribution of the videos to cache servers that act of proxies for clients without these specific features.

3.5 Conclusions

In this chapter, we have examined techniques that we have derived from *patching*, which is a TVoD descendant of family of NVoD techniques derived from *skyscraper broadcasting*. We have presented our optimization options for patching and we have presented a TVoD variation of this family that increases the scalability and stability of these systems by introducing cache servers. We have called our approach *Gleaning*.

Gleaning can benefit from all kinds of optimization options that can be applied to patching. From our optimization steps, we have drawn several conclusions. The most important one is that the patching window size should not be determined by the receiver but rather by the

sender, which has more information about the overall popularity of a movie title. This is important since optimizations of the window size are also optimizations of the number of concurrent streams that need to be supported by the server. Differences in local user community behaviour will affect primarily the cache server and is handled by its removal strategy. The centrally controlled decisions of the origin server, on the other hand, influence the load of the distribution system.

We have further found that *multilevel patching* increases the savings in server bandwidth, but is a lot more complex to implement. Furthermore, the number of concurrent streams that are received at the client resp. at the cache server increases. It seems appropriate to implement single-level patching only or at least to limit the number of concurrent streams that are used.

Another limit of the patching technique should not be ignored in implementations. For a sufficiently large user population and the top-popularity movies, it is appropriate to apply a simpler technique such as batching; Depending on the user acceptance of delays, the delivery of a movie can be delayed by several seconds, reducing the management overhead of patching. For each patch stream, a batching window of similar size can be applied - this allows for additional joining, and especially in conjunction with multilevel patching, limits the total number of levels that must be supported concurrently.

Another issue is the use of caches in wide-area distribution systems. The network usage, which becomes relevant in large-scale and wide-area systems, is not considered in broadcast approaches, since those aim either at smaller scale systems, such as metropolitan area networks, or at systems with real broadcast media such as satellite distribution. The infrastructure for such a distribution system needs (a) immediately deployment to a large initial user population, and (b) is not smoothly scalable to larger areas.

Our use of caches is also important for the stability of the overall system and for the ease of deployment and scalability of the system. The pure financial gain of a cache-based system may not exceed a centralized architecture sufficiently by itself, i.e. without the stability considerations. Another issue with the use of caches is the possibility to implement write-through caching and to support stupid clients. Such a possibility would allow a deployment with a reduced upgrade of the public infrastructure that is already in place.

The negative aspects of gleaning as a distribution mechanism are moderate compared to these gains. It requires a reliable multicast technique, especially when multiple caching levels are used and errors could be accumulated in transfers among caches. We must also prevent that malicious senders trash the distribution networks by indicating to their clients a need for buffers that are too large.

The caching itself is an issue that is mostly unaffected from this proposed distribution mechanism. Replacement strategies in the caches remain entirely unaffected by the distribution mechanism. Caches will join the streams that are delivered from the server, which implies that the caches can decide whether a content is cached or not. Consequently, caching strategies can vary in gleaning. For example, this allows the combination of gleaning with *hint-based caching*, since only removal strategies are affected by the hints. Furthermore, gleaning demands that a cache sets aside an amount of storage for each video stream that is delivered to a client at a given time. This gives leaves three options to the cache implementation:

- an always-overwrite approach, which requires the complete storage of the full movie at least for some time
- a conditional-overwrite approach with a fixed amount of storage space set aside for temporary use as buffer for movies that are not considered relevant enough for caching
- a conditional-overwrite approach that shares storage between fully cached movies and cyclic buffers dynamically; this approach is similar to [AlAm96], where movies receive larger amounts of buffer space on routers, depending on the number of concurrent hits. However, this setting favors smaller buffers for more popular titles, unless they are worth being cached completely.

After these investigations of the distribution system, we will consider other aspects of the distribution system before it is evaluated as part of a complete system.

4. Protocol Suite

Since the goal of this thesis is the examination of missing links for a decentralized video distribution system, existing protocols to support this must be evaluated and if necessary, new protocols must be defined. Especially in combination with the requirements of the patching and gleaning approaches, protocols have to be reconsidered since these approaches have been introduced only recently, resp. are introduced with this thesis. The fact that this chapter precedes the investigation of caching strategies is based on the following:

Protocols are meant for data distribution. They are independent of specific decisions concerning replacement algorithm or distribution techniques.

This opinion has developed during the concurrent work on server and protocol implementation and the evaluation of caching policies. Although it is beneficial if distribution techniques and the caches' removal strategies fit well, both will be operational in most combinations, although the performance will degrade in some combinations. We consider this assumption basic, since we have observed that this is typically approached as a monolithic problem in existing distribution system. The assumption allowed to work on protocols while optimized replacement mechanisms were not fully investigated.

Based on the requirements that are deduced from the previous chapters in Section 4.1, the existing control and data transfer protocols are evaluated in parts Section 4.2 (stream control), Section 4.3 (video streaming) and Section 4.4 (reliable multicast). In the decisions that were made for the definition of our complete system, we have taken into account that some of the discussed protocols are more difficult than others to install for use with a wide-area distribution system, when only a minimal functionality extension needs to be achieved. Another decision that was made in preparation to our system design is that we intend to achieve compatibility with current Internet mainstream protocols. This is a decision that is not necessarily the outcome of a commercial implementation of such a distribution system; alternative protocols may be based on other standardization work or even on proprietary protocols. The protocol suite that we design with these goals in mind is introduced in Section 4.5, and protocol elements that are new in our protocol specifically are described in Section 4.6. Finally, Section 4.7 evaluates this step.

4.1 Requirements

The distribution system that we are envisioning imposes several requirements on the protocols that are used in such a system. The problem with the collection of these requirements is the uncertainty of the features that need to be supported by the protocols. To add structure to these requirements, we distinguish general requirements, which are required of all protocols that are applicable for our kind of distribution system, generally convenient features, which are useful for many applications but not always necessary, and specific requirements, which are only necessary for some distribution mechanisms. The requirements have been defined according to the requirements of the distribution systems that we envision; they should be independent of the information that is interchanged between caches to increase the performance of caching decisions as well as of the replacement decisions that are implemented in the caches.

4.1.1 General requirements

General requirements on the protocols are independent of specific decision concerning the distribution mechanism, and certainly independent of the caches' removal strategy. However, this thesis examines the distribution with caches specifically, which imposes some general requirements.

Separate control and data protocols

The separation of control and data protocols is a principle approach that has been implemented in Internet video streaming protocols for years, without much consideration about the reasons. Certainly, DSM-CC is multiplexed in MPEG-2 transport streams, but in on-demand systems this is usually a multiplexing step that is independent from the video stream. The amount of feedback about the stream quality that is transported with the data stream differs from one protocol to another, sometimes stream setup and QoS negotiation are handled in-band with the data stream, but control information, such as stream location is exclusively transferred out-of-band.

Recently, the term "HTTP streaming" has been coined[1]. Basically, this is an HTTP GET request for a video file, but the server can draw conclusions about the client actions from the behavior of the TCP stack; this can be considered *implicit signalling* of the control information. It allows the server to determine Start, Pause and Stop actions, and it allows scaling of the content based on the throughput that is experienced at the sender side. The use of TCP makes it unscalable, but with a different transport protocol, it may be.

We have decided not to work on the latter approach. First of all, the separation of control and data protocols allows the adoption and adaptation of existing protocols. The second reason is that it is also technically favorable because of its modularity. Besides, multiplexing at the network level is always possible, as demonstrated by MPEG-2.

Reliable data transfer to caches

For usual MBone-conferences with tools like vic [MJ95] and vat the functionality of RTP is sufficient. As video- or audio streams are transmitted and displayed continuously, small losses within the information are of minor significance. It would be more complicated to retransmit lost data, because they may disturb the normal procedure. With respect to a video-transmission the pictures would be displayed incorrectly and the audio be distorted. But there is a difference in using unreliable transfer between video cache servers. A cached version of the movie on a cache server should be stored 100% correctly to avoid error propagation towards the client. With the use of standard streaming protocols, information that gets lost during transmission is also lost to the caches. The problem is that these errors would be transmitted with every stream that is forwarded from the cache server to a client. This should be avoided since it has to be regarded as a degradation of the service quality. The amount of errors would be rising in a scenario where movies are distributed in a multi-level hierarchy as well, by being stream-transmitted from one cache server to another one that is located further downstream from the library server. During each transmission data can get lost and thus lead to a higher error rate in stored

1. The term is used in Real Systems product brochures but they are probably not the original source.

copies. We consider this unacceptable and require reliable transfer to the caches in our protocol.

Redirection support

The requirement that a protocol needs redirection support at the service level is trivial for a distribution system that is based on caching. Without support and permission for redirection, the original server would not be able to receive requests from client through a proxy cache server. The use of caches would still be possible, but this would require the implementation of the functionality of redirection in a different layer of the application.

Potentially, this could be the IP layer, such as network address translation as proposed in [RFC2391], or it could be implemented in the application itself. The primary drawback with the first approach is in our view the need for central control. The second control would be a substitute but basically, an additional control protocol that could work more efficiently if it were integrated with the other control elements.

Support for caches that are not routers

Caches are often implemented as system modules that must be passed by all content in order to reach the requesting client. This has been typically the case for CPU caches (there are exceptions, e.g. [SoLe97]), and it is typical for web caches. Some web server products are even installed on routers (e.g. [Cisco]). This contradicts our assumption that efficient video caching will probably be achieved by strategies that apply conditional caching of videos.

Neither can we assume that a video server is a router at the same time; typically we can not even assume that the network service provider is the cache owner at the same time. Thus a protocol should be capable of dealing with cache servers that are not located on the default delivery path between the original server and the client.

4.1.2 Generally convenient features

Features are considered advantageous for most protocols that could be implemented in a distribution systems. In contrast to the general requirements, they are not necessary for the operation but they could save resources or work.

Data multicast support

Multicast support is generally convenient for video distribution as it can reduce the bandwidth that is required for video transmission from one server to multiple client considerably. Especially for conferencing systems, this is an asset. In conjunction with video-on-demand, this has not been used initially, and techniques for the combination of requests into “batches” had to be introduced first. These approaches were presented in Section 2.5. With caching, the application of such techniques may be considered unnecessary. We do not believe that this function should be neglected since even caches that are accessed by a small user community could benefit from batching or similar techniques when highly popular titles are served. In order to support this, multicast is mandatory.

Segment concatenation

Some distribution systems require support for the concatenation of content segments. The clients receive pieces of the content from different sources, caches receive pieces of the content from different sources, or ideas such as content insertion ([VeLi95]) are applied. This could be handled transparently by the cache servers, by re-encoding or by re-multiplexing the content before it is delivered to the clients. Such an approach would have the two negative aspects that (a) the server load may be increased considerably, and (b) all traffic would have to routed through the cache server, although the cache server may have decided to redirect the request rather than store a copy of the movie itself and although the client may be able to receive a multicast stream from the original server directly. Thus, segment concatenation would be an advantageous feature for a protocol suite in our envisioned distribution system.

4.1.3 Specific requirements

Specific requirements can not be applied generally to video distribution systems. They are either specific for our intentions, or they are specific for distribution approaches that we investigate.

Internet protocol

There is not technical reason for this requirement, but Internet protocols have been used over most infrastructures successfully. Furthermore, we have the Internet readily available, the Internet standards have always been freely available, and we are more experienced with Internet protocols than any other protocols suite.

Caching prevails over multicast

On demand-systems are frequently operated in plain on-demand mode, i.e. without any application of multicast. Even in such an application, our protocols should work properly and implement the decisions that are made concerning an appropriate caching strategy. We require multicast for other reasons, but the support for caching takes precedence when there is a conflict in the protocol design.

Support for patching

The catching idea of Chapter 3 should be supported; this requires support for request redirection and for concatenation of stream segments if an efficient implementation is intended. Specifically, the concatenation of stream segments must be supported at the client side, if we do not want to implement this service from the cache server. The latter approach could be taken, but the above-mentioned general requirement that cache servers should not be routers contradicts this approach.

Support for write-through mode

Do this potentially from different files on different servers. This is not an issue for us because we can operate in write-through mode: if the cache joins a multicast stream too late and we

apply catching, the client listens to the Patch stream that is delivered from a higher level server first, and receives the remaining part of the video from its cache server

A complete protocol suite is required to co-operate for the implementation of a video-on-demand system. Completely specified architectures such as DAVIC and DVB can hardly be replicated in a university scenario, but - as noted in the Introduction - Internet VoD is a growing market, and it is based on protocol specifications that are freely available and, independently from each other, of very limited complexity.

4.2 Stream Control

Since the beginning of streaming media, a large number of control protocols and languages have been implemented and used by the researchers and also in the earlier products. While the relevance of interoperability at the bitstream level has been recognized early, and soon after that of protocols for data transfer, the relevance of control protocols has largely been neglected. A suggestion of this negligence is made by the application of QoS mechanism to content distribution without applying QoS mechanisms to the control channels.

In the recent past, two approaches to stream control have been standardized by ISO and by IETF. Since then, standards as well as commercial products make use of these approaches to achieve interoperability.

4.2.1 Distributed Storage Media Command and Control

The Distributed Storage Media Command and Control (DSM-CC, [ISO96]) is a part of the MPEG-2 standard. It consists of two parts, the User-to-User part (UU, part 6 of MPEG-2) and the User-to-Network part (UN, part 7 of MPEG-2). The UN part specifies the communication of application and network services for resources, which is not the issue of this section. The goal of the UU part is the specification of generic multimedia interfaces that allow client applications service access in a platform-independent way. While the specification of UU includes definition of issue such as data types, a common API, and the user environment, mainly the functionality is relevant to this section.

To the client, UU appears as an API. i.e. client programs are implemented like applications that perform remote control over a service. These services include stream operations, file operation as bulk data, directory operations, session operations as well as communication with service gateways to access services of provider other than the immediate service provider. Extended interfaces provide download functions, subscription to events, viewing and sorting of server-side objects, authentication, versioning of objects, or configuration of the communication mechanism itself. The API is described in IDL and uses Corba for communication.

Presentation Description

Presentation description is not an issue with DSM-CC; DSM-CC’s approach to this is the download of environment-dependent applications that behave are understood by the end systems. An application of this approach is the use of MHEG in the DVB system and in DAVIC.

Synchronization

The typical environment of DSM-CC is an MPEG-2 based delivery system. In such a system, it is typical that the application server creates an MPEG-2 multiplex which guarantees the synchronized delivery of media. Especially for applications that include uni-directional content distribution such as satellite television, this is augmented by the concept of an object carousel, which is a FIFO object cache at the receiver side.

4.2.2 Real Time Streaming Protocol

The Real Time Streaming Protocol (RTSP, [RFC2326]) is an IETF RFC that is supposed to be used in conjunction with various other protocols. Its functionality is not generic but rather concentrated on stream control. It references elements of HTTP to which it is weakly related. It can be used with either TCP or UDP as an underlying transport protocol. The data transfer protocol that is mentioned in the RFC and that interacts most closely with RTSP, is the Real-Time Transfer Protocol (RTP). The same approach applies for the session description protocols; although no fixed session protocol is defined, the RFC specifies the interaction with the Session Description Protocol (SDP).

The protocol is a text-based protocol that refers explicitly to HTTP in parts of its descriptions, and actually it includes several directives from HTTP instead of redefining them. The functionality added in this way includes proxy-support and authentication.

Presentation Description

The Session Description Protocol (SDP, [RFC2327]) is originally considered as a companion protocol for SAP, the Session Announcement Protocol. However, besides this mode of distribution for session information, others like download from the web or E-mail distribution are also compatible with this kind of information. Basically, SDP provides a line-oriented syntax to describe a multimedia session in ASCII.

Synchronization

The Synchronized Multimedia Integration Language (SMIL, [h:Hos98]), is RSTP's preferred approach to deal with distributed multimedia presentations that require synchronized presentations of individual streams.

4.2.3 HTTP Streaming

A straightforward application of HTTP has been used for the control of real-time streaming as well. With this approach, which does not control the stream at all except for implicit signalling of request, congestion and stream end, it is at least possible to present video in a straight-forward way. To present video clips for advertisement purposes, this has been proposed as a solution which is applicable at sites with very limited requests.

4.3 Video Streaming Protocols

This section shows that the number of existing video streaming protocols is large, and than presents reasons for the selection of RTP as an element of our protocol suite. The handling of the

'competition' is rather short for the importance of the protocol selection for an implementation. However, the main goal of this thesis is the study of feasibility of complete systems. Certainly, the modifications that are necessary to achieve the functionality that we require could also be achieved with a different protocol as a starting point.

4.3.1 Collection of Internet Approaches

The protocol-oriented approaches of the Internet have been manifold, and they have been implemented at various levels of the IP protocol stack. Because of the number of approaches, only a short explanation is given for each of them.

- Plain UDP has frequently been used for straightforward transmission of packetized video in LANs.
- HTTP-Streaming is essentially TCP; it has been mentioned above.
- A multitude of pure ATM approaches; however, I do not believe that ATM is going to become an exclusive quasi-standard for end-to-end transmission of video.
- The transfer of MPEG2 streams has been specified over many means of transport, including specifications by DVB and DAVIC for the use of CableTV network or satellite, over ATM, over IP over ATM, or as an RTP payload.
- XTP (Xpress Transfer Protocol, [SDW92]) was a competitor of layers 3 and 4 of the Internet protocol stack. The central intention was the development of a standard with support for generic service that could be selected in arbitrary combinations by the application; including multicast and QoS negotiation.
- ST-II (Stream Protocol 2, [RFC1819]) was a multicast protocol with QoS support at the network layer (an IP companion). It was used with HeiTS (Heidelberg Transport System, [DHH+93]) as a transport protocol, or with partial XTP functionality as a transport protocol (called Berkom MMT or XTP-Light, [h:SaDe94])
- IPv6 has added a so-called flow id. This allows out-of-band QoS negotiation for flows. Using these reservations, higher level protocols such as UDP can then make use of the reservation by sending IPv6 packets with that flow id.
- IntServ (Integrated Services, [RFC2205]) is a receiver-oriented out-of-band signalling approach for dynamical QoS negotiation in multi-party communication. It allocates resources to receiver/stream identification but can also work with IPv6 flows.
- DiffServ (Differentiated Services, [RFC2474], [RFC2475]) is a point-to-point virtual leased line approach that allows service providers to interpret IPv4 ToS bits or IPv6 labels in a consistent way to provide QoS. The means to negotiate and to guarantee the service are not specified yet.
- A variety of research prototypes have applied layered transmission (e.g. [AMK97]).
- A multitude of proprietary protocols over IP and UDP has been used specifically for video distribution or conferencing - e.g. Apple QuickTime before version 4, Real Systems' SureStream ([h:Real99]), the original Vosaic VDP (Video Distribution Protocol, [CTC+96], now VTEL uses H.323), or its research successor MSP (Media Streaming Protocol, [Hes98])
- RTP (Real-time Transport Protocol, [RFC1889]) is developing into the quasi standard for video packaging in the Internet. Since we use RTP, it is described in detail below.

4.3.2 Internet Quasi-Standard

The Real-time Transport Protocol (RTP) was created to transport real-time data over the Internet. The first thing that needs to be noted is that it is neither real-time, nor is it a transport protocol. It is an application-level framing approach that allows applications to exchange information about the stream quality.

Originally the Internet was created to transport non real-time data belonging to applications like telnet, E-mail, ftp. The early Internet development was funded by the military and required problem resistance rather than performance. The early applications require correct and complete data transmission without any time restrictions which is given by the TCP/IP protocol. TCP ([Pos81]) for example has mechanisms to guarantee the correct, complete delivery of data. In contrast to this VoD or other real-time applications make specific time restrictions on how the data is delivered. Internet telephony, MBone-conferences and all video- and audio conferences can not or not satisfactory be realized with the usual protocols. RTP provides functionality to realize real-time applications, but it does not provide any time QoS (Quality of Service) guarantees. QoS guarantees have to be provided through underlying protocols like for example RSVP ([BZB+97]). RTP provides payload type identification, sequence numbering, time-stamping, delivery monitoring and supports multicast if the underlying protocol provides this service.

RTP is a protocol independent format to transmit real-time data. Usually it is used over UDP (User Datagram Protocol, [Pos80]), as UDP allows multiplexing and does not have any retransmission schemes like TCP. A protocol dependent retransmission mechanism would probably violate the time restrictions. RTP is used together with RTCP (RTP Control Protocol, [RFC1889]) which allows a quality monitoring of the network connection and has minimal control over the session. Furthermore RTCP can be used to identify the sender. The main task of RTCP is to send periodic control packets to all members of the session using the same distribution mechanisms as the data packets.

Favorable for RTP is also, in opinion, the increasing support by public domain as well as commercial tools. The following tools and systems have been implemented with RTP initially, or have been modified to use RTP in their recent version: Apple QuickTime, IBM VideoCharger, SUN's Java Media Framework, the MBone tools, Cisco IP/TV.

Also DAVIC, which has up to now (specification 1.4) always referred to MPEG-2 delivery over broadcasters' traditional end-to-end infrastructures, is working on the additional support for Internet protocols in specification 1.5. There original approach for an integrated Internet access meant the delivery of IP embedded into MPEG-2. Now, the delivery of audio-visual material using 'native' Internet tools is considered. Early versions of the specification are currently available. The draft of part 4, which was published in May 1999 and is still rather unspecific, lists RTP and RTCP for stream delivery. It proposes the use of RTSP and SDP[2] for session control and session description, and the use of the Service Location Protocol (SLP, [RFC2608]) for service location. The document refers to the Resource Reservation Protocol (RSVP, [RFC2205]) for optional resource reservation and is also referring to routing protocols,

2. via SAP, HTTP or E-Mail

transport and network level protocols that are necessary to build a complete delivery infrastructure.

4.4 Reliable Multicast Protocols

The design of a reliable multicast protocol is determined by the requirements of a specific application or area of applications that the protocol is built for. Different applications impose different requirements on the underlying reliable multicast protocol. Possible classifications of multicast protocols can be made by the type of error recovery and the ability of transmitting real-time data. [h:WCW99] defines two types of error recovery: Centralized error recovery (CER) and distributed error recovery (DER). CER allows retransmissions only to be performed by the multicast source. DER allows retransmission to be performed by all multicast members having the correct data. The suitability of the protocol to transmit real-time data depends on how the data is recovered. Real-time applications will accept a lossy data flow but they will not accept a significant delay. This implies that data recovery should not interrupt the flow. An example for an application that accepts lossy data flows but can not handle retransmits very well is a video conference system. If a gap is detected, it is better to display the subsequent data instead of pausing the stream, waiting for the lost data and than continue with the play of the data. Other applications like a white board conferences may require a delayed repair while displaying the currently available, outdated data.

Some examples for reliable multicast protocols are SRM (Scalable Reliable Multicast) [FJL+97], TRM (Transport Protocol for Reliable Multicast) [SBD96], RMTP (Reliable Multicast Transport Protocol) [LiPa96] and LRMP (Light-weight Reliable Multicast Protocol as an Extension to RTP) [Lia98]. SRM and TRM are DER type protocols and LRMP and RMTP are CER type protocols. TRM and LRMP make similar assumptions about loss detection and repair requests as SRM, so SRM can be discussed as an example for all three protocols. RMTP provides sequenced lossless delivery of bulk data (e.g. Multicast FTP), without regard to any real-time delivery restrictions. It uses a windowed flow control and ACKs for the received packets. This technique allows a reliable transmission, but if packets are lost, the data flow is interrupted because the lost packets are resent immediately by the sender which leads to a non-continuous data stream. So this protocol is not applicable for VoD applications.

SRM is a reliable multicast framework for light-weight sessions and application level framing. It's main objective is to create a reliable multicast framework for various applications with similar needs of the underlying protocol. SRM does not distinguish senders from receivers. Whenever data is created, it is multicast to the group. Each member of the group is then responsible for loss detection and repair requests. The repair requests are multicast after waiting a random amount of time, in order to suppress requests from other members sharing that loss. Every member capable of sending a repair packet also sets a timer and if no repair packet is sent from another member it sends the repair packet. After sending this packet a new timer is set in order to avoid any possible duplicated requests from the receivers. This mechanism tries to suppress duplicated retransmission requests and duplicated repair packets. As it is possible that the last packet of a session is dropped, every member multicasts a periodic, low rate, session message including the highest sequence number. How to compute the time for the timers is discussed very precisely in [FJL+97]. SRM was tested and implemented in *wb*, a white

board application for real-time conferences. It must be mentioned that SRM needs a specific distribution infrastructure which is not widely available in the Internet at the moment.

A third class of reliable multicast protocols are the ones which include FEC (forward error correction) as a technique to achieve reliability [NBT97]. Reliable multicast achieved through FEC is also applicable for VoD systems, since usually no retransmissions are necessary during the multicast transmission of the video stream. The major drawback of this approach is, that error correction information appropriate for the client with the worst connection must be included in each multicast packet. This will lead to a higher use of bandwidth thus leading to a reduced connection quality for the clients. In addition a completely new protocol must be built in the case of layered FEC since this model is not compatible with already existing protocols.

All of these existing solutions have been taken into account. They have been considered in relation to the one protocol that is currently used for streamed transmission to end-system in the Internet domain is taken into account as well, which is RTP, the Real-time transfer protocol. All of the above approaches to reliable multicast suffer from one common problem, besides potential other problems, and besides their benefits: they are not compatible with RTP.

Since players of commercial video-on-demand systems which would benefit from a video distribution infrastructure as envisioned by this thesis are typically working with RTP-compliant receivers at the client side, an RTP-compliance in the distribution system would be beneficial. Therefore, we have implemented an alternative that fulfills our requirements based on RTP, which is described in the following sections.

4.5 Selected Protocol Suite

The protocol suite that we are proposing in this section is selected in this way due to our goal of RTP compliance, and interoperation with existing tools and protocols. In spite of this goal, we want to be able to support the optimization ideas for caching and distribution systems that are not supported by current implementations. This demands new protocols that maintain backward compatibility but include our requirements, which we call LC-RTP and LC-RTCP. The protocols implement the functionality for the requirements that have been listed Section 4.1, and allow experiments for several variations of the distribution system and removal strategies. Concerning the other protocols that are include in the complete suite, RTSP and SDP, we did not need to modify the protocols themselves; however the session description that we are distributing may not be considered trivial.

The protocols are intended to unload stream transmission effort from the servers, routers and networks, while an increase in the necessary effort for the control of the system is acceptable. We assume that the control server is probably powerful enough to handle a few transactions that are necessary to manipulate the control server.

4.5.1 LC-RTP

RTP with Loss Collections (LC-RTP) implements our idea of a unified protocol for stream transmission that is compatible with RTP, and reliable transfer of content into the cache servers. It solves these problems by making RTP reliable, while the ability is maintained that non LC-RTP capable clients (standard RTP clients) can receive an LC-RTP stream as well.

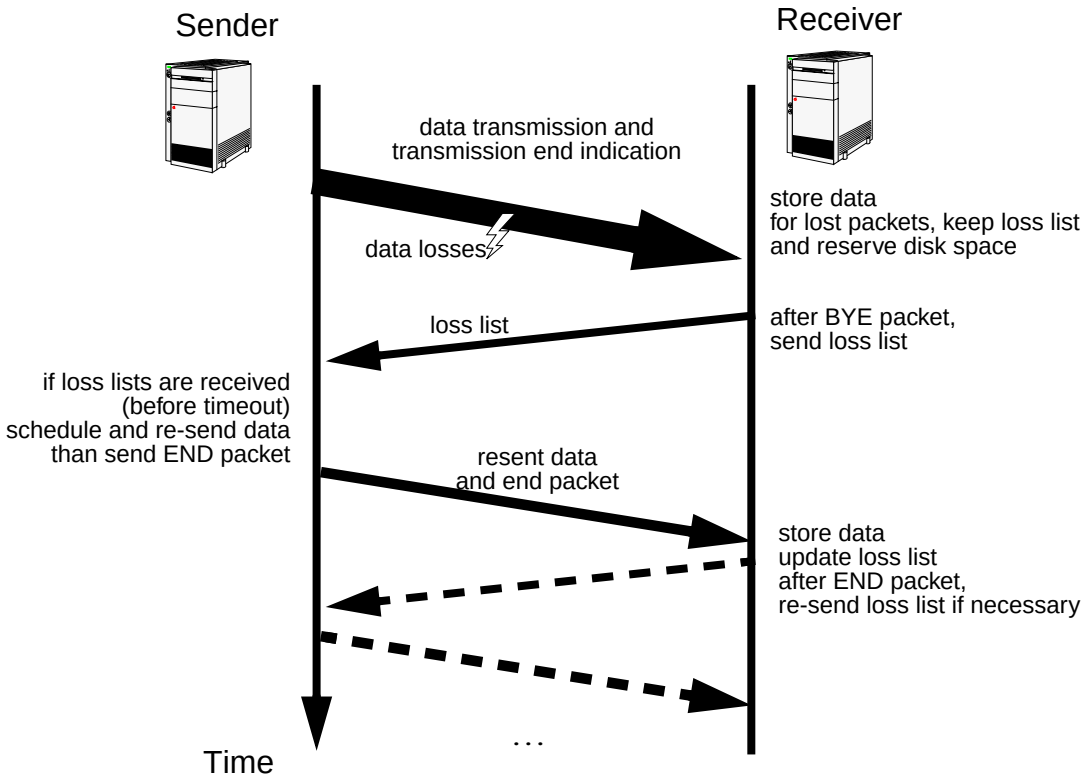


Figure 18: LC-RTP Communication

To describe LC-RTP the transmission process is divided into two parts. The first part works like a regular RTP transmission and ends when end of the movie has been transmitted (using the BYE message). The second part follows this BYE message and is used to retransmit all lost data. In this scenario all receivers that are cache servers that have decided to keep a movie in the cache, and that have experienced packet loss, will continue to receive packets after the RTP BYE message. Figure 18 gives a general overview of the different steps that are executed during a LC-RTP session.

4.5.2 LC-RTCP

Just as RTP has a companion protocol RTCP for the exchange of information about the data transfer, LC-RTP requires a companion protocol LC-RTCP, which needs to be RTCP-compliant. In application-defined RTCP packets, the receivers inform the sender about their losses after the reception of the BYE packet, unless all of its missing packets have earlier been reported by another receiver.

4.5.3 SDP

The Session Description Protocol (SDP) has been produced by the MMUSIC working group of the IETF. It was originally intended as a complement for the session announcement protocol SAP to communicate conference addresses and tool-specific information over the MBone. Alternatives such as HTML postings or E-mail distribution of session descriptions were taken into account as well. With this primary goal in mind, SDP does not support negotiation of any of session information, but is just used for dissemination.

With the exception of character encoding rules, this line- and column-oriented protocol is extremely simple. Table 5 shows all of the two character keywords of SDP in the exact order of occurrence in a session description. Keywords must be in first column of a line, without whitespace before or after the equal sign, and are followed by a set of values on the same line. Carriage return and newline characters determine the end of line, without escaping options.

keyword		meaning	occurrences
v=		protocol version	1
o=		owner/creator and session identifier	1
s=		session name	1
i=		session information	0-1
u=		URI of description	0-1
e=		E-mail address	0-1
p=		phone number	0-1
c=		connection information	0-1
b=		bandwidth information	0-1
time description block			>=1
	t=	time the session is active	1
	r=	zero or more repeat times	0-1
z=		time zone adjustments	0-1
k=		encryption key	0-1
a=		zero or more session attribute lines	0-1
media description block			>=0
	m=	media name and transport address	1
	i=	media title	0-1
	c=	connection information	0-1
	b=	bandwidth information	0-1
	k=	encryption key	0-1
	a=	zero or more media attribute lines	>=0

Table 5: SDP protocol format

We have found SDP appropriate without changes for our purposes. For that reason, this section is restricted to a demonstration of SDP's applicability (in conjunction with RTSP) to the complicated case that the patching mechanism is applied transparently to the clients at the caches.

The movie in MPEG system encoding is requested on Oct 17 17:54:46 (3149164486), and it runs for 90 minutes, i.e. until 19:24:46 (3149169886). This initial viewer will receive the ses-

sion description of Figure 19. The encoding format, RTP/AVP is supposed to deceive the client that understands only RTP. The only deviation from a regular RTP transmission that would be announced by a server is the session attribute *fmtp:lcrtp*, which indicates to the cache servers that our proprietary protocol extension is used as well. Note that the media attribute *rtpmap* is only necessary due to a historical incompatibility of the VideoCharger, which sends MPEG1 streams with an encoding format value 0.

```
v=0
o=vsadmin 3149164486 3149164486 IN IP4 192.168.2.1
s=phantclip.mpg
i=The Phantom Menace
c=IN IP4 224.2.24.8/16
t=3149164486 3149169886
k=prompt
a=recvonly
a=fmtp:lcrtp
m=video 49170 RTP/AVP 0
a=rtpmap:0 MPEG1/1411200
```

Figure 19: SDP specification for an initial LC-RTP stream

Another user will request the same title five minutes after the start of the movie, i.e. at 17:59:46 (3149164786). When its proxy cache communicates with the original server, it will receive the session description of Figure 21. This session description contains two time fields, the first giving the original time span, which has already started. The second is the display time of the patch stream, fives minutes from the current time. In the first media description block, information is given that allows to join the multicast stream; in the second media description block, the batch stream is described. It is sent with port information that differs from the original port. This is necessary to allow pass-through delivery of the initial portion of the movie to the client - the packet sequence numbers of the main portion of the movie, which are higher than those that it expects, would force the client to assume major packet losses in its session.

```
v=0
o=vsadmin 3149164486 3149164786 IN IP4 192.168.2.1
s=phantclip.mpg
i=The Phantom Menace
c=IN IP4 224.2.24.8/16
t=3149164486 3149169886
t=3149164786 3149165086
k=prompt
a=recvonly
a=fmtp:lcrtp
m=video 49170 RTP/AVP 0
a=rtpmap:0 MPEG1/1411200
m=video 49172 X-LCRTP/AVP 0
a=rtpmap:0 MPEG1/1411200
```

Figure 20: SDP specification for an joining LC-RTP streams

In case of support for Patching or for a variation of Patching (such as the Catching approach described in Chapter 3 of this thesis), it is necessary to support segmented streams and partial retransmission. To support this, another request is re-routed through an LC-RTP-capable proxy server.

The cache server needs to reconstruct the SDP description. Figure 21 shows how the example is modified to include the information that the proxy server is giving to the client to implement a concatenation of the patch stream and the cached stream into a contiguous sequence of a longer one. In this modified SDP description, several details are of interest:

```
v=0
o=vsadmin 3149164486 3149164786 IN IP4 192.168.2.1
s=phantclip.mpg
i=The Phantom Menace
c=IN IP4 224.2.24.8/16
t=3149164486 3149170186
k=prompt
a=recvonly
a=fmtp:lcrtp
a=control:rtsp://cache.server.com/phantclip.mpg
m=video 49172 RTP/AVP 0
a=rtpmap:0 MPEG1/1411200
a=control:patch=1
a=range:npt=0-360
m=video 49172 RTP/AVP 0
a=rtpmap:0 MPEG1/1411200
a=control:base
a=range:npt=360-324000
```

Figure 21: Pass-through SDP specification moving from the proxy cache to the client

- the *t=* field is now showing start and end times that cover the complete movie length with a time offset appropriate for the 5 minutes that the client has arrived after the original start,
- the *a=fmtp:* line is kept for informative purposes
- the session level line *a=control:rtsp://cache.server.com/phantclip.mpg* indicates that aggregate control is being used; this is necessary and must be enforced by the proxy cache. If the client would be allowed to manipulate the video sessions independently, the situation may arise that the second part of the movie is displayed in parallel with or with an offset from the first part.
- the media level lines *a=control:patch=1* and *a=control:base* are server-chosen names for the stream elements that are delivered.
- the lines *a=range:npt=0-360* and *a=range:npt=360-324000* imply for the client that the second stream needs to be played in sequence with the first one.

4.5.4 RTSP

We have used RTSP as the one control protocol that is currently replacing proprietary control protocols from the Internet applications. In theory, based on the study of the RFC ([RFC2326]), this protocol should solve all of our requirements for a control protocol if it is jointly with SDP in an appropriate manner.

We have implemented RTSP client code based loosely on the Real Networks demo client code (am RTSP 0.6 implementation), and we have implemented the RTSP server code at a time when no freely available code such as that of the PRISS server ([h:Stre99]) was heard of. We have tested our code with several commercial servers and have experienced compatibility as well as semantic problems. Examples of implementation problems are listed first:

- Implementations are restricted to a (proprietary) interpretation of order and session information.
- Implementations follow different versions of the standard.
- Client identification seems to refer to processes, which may be a replacement of the missing user identification on single user client operating system - starting a new session requires killing of the process.
- The mapping of request to files on disk (resp. to assets) could not be determined
- The interpretation of the application header could not be determined.
- The interpretation of the session description header are unclear and experiments have non-deterministic effects.

Semantics problems concern interpretation of (or deviations from) the standard:

- Implementations' handling of non-responsive clients: how long should the transmission to the client be continued? (the RFC proposes a 60 seconds timeout period)
- Implementations' handling of client crashes: When the client returns to the network with a different DHCP address, how to identify? (the RFC proposes to accept session ids and authentication as sufficient)
- Teardown semantics of a TCP session: a TCP session could be able to survive a control channel close, but how long does the server keep the session state if there is no reconnection? (the RFC proposes not to keep the session when a persistent connection shuts down)
- Teardown semantics of a UDP session: how can the server recognize a client crash/restart?

In spite of all these problems, we have decided that the RTSP RFC is an appropriate specification of a control protocol for our goals. In conjunction with an SDP interpretation as the one presented in the previous section, we can address our requirements with an own implementation.

4.6 Operation of LC-RTP and LC-RTCP

This chapters presents our protocols LC-RTP and LC-RTCP in detail. First, the protocol operation is explained by showing the actions of senders and receivers in the regular transmission phase and afterwards, in the retransmission phase. Section 4.6.2 specifies LC-RTP, based on the RTP specification, and Section 4.6.3 specifies LC-RTCP. Section 4.6.4 presents results that were made during the various tests of the protocol, which show its applicability for use in a wide-area on-demand scenario.

4.6.1 Design

The design of the protocol is derived directly from its intended operation. As a protocol that operates in two separate phases, transmission and re-transmission, with different requirements, we explain the design by presenting the protocol actions in these two phases. The first phases, transmission, is supposed to be RTP-compliant. The second phase, retransmission, has no such requirement.

Actions during the movie transmission

- SENDER

The sender streams a movie which is requested by a client as a multicast stream to all receivers of a multicast group that includes that client. In order to give the receiver the possibility to reserve exactly the required disk space in case of data loss, it is necessary to send information beyond the regular information of an RTP packet. In our case this consists of a byte count. The sender calculates a byte position of the RTP payload, given as the relative position to the stream start, and transmits this information with the data in an extension of the RTP header. A connection between the byte count and the file position of the stored movie is not always necessary but can increase cache performance in conjunction with an appropriate buffering strategy or file system.

If possible the byte count should be included in the packet, because it facilitates the synchronization between byte count and the data which are represented by it. For example, if the byte count is sent in an extra packet, or via RTCP, the sequence of the byte count and data packet can be changed, or the byte count packet can get lost. If the receiver receives only the data packet, it does neither know whether any data is lost nor how much data is lost. Thus, it is not possible to write the data to the file without buffering large amounts of data or alternatively, without risking time-confusing repair steps in a later repair phase, because there is no information at which position the data should be written in the file.

The byte count can be implemented by as offset-list. By comparing the byte count with the file position of the portion of data that has already been received, exact loss information can be stored in the offset-list. When the sender receives the message of losses, the offset-list can be mapped to the file. If the byte count is equivalent to the number of bytes of RTP payload that has been sent through the network, an encoding-independent storage format can be realized. As a consequence it is possible to have different file layouts on the sender- and receiver side. Each cache server implementation has to transform the mapping of the byte count into its own format. For example one cache server implementation stores the file as raw data and another stores some header information with it.

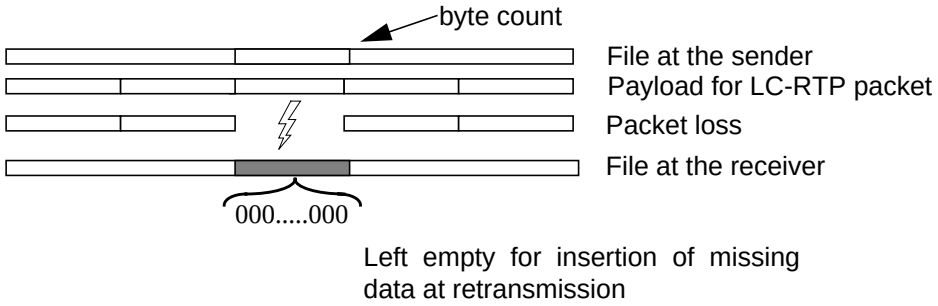


Figure 22: LC-RTP byte count supports retransmission

As a consequence of including the byte count in the data packet, and the requirement of servicing regular RTP clients, only an RFC-conforming protocol extension was an option for us; including the byte count in the payload of the packet would cause problems for standard receivers, like most of the clients are.

At the end of the movie transmission, an end packet is sent including the last byte count, in order to inform the receivers of the normal end of the transmission including information to check whether data preceding the end packet was lost. With this end packet the sender has transmitted a whole video as a multicast stream.

- RECEIVER

The receiver stores the data and detects a loss by checking the byte count with the last memorized byte count. If a packet loss is detected, the difference between the two byte counts and the length of the actual packet is computed and this computed size can be reserved on the disk for a later insertion of the retransmitted data (see Figure 22). The received payload of the packet is then stored after this reserved gap. Furthermore the loss must be written to a loss list. If no loss is detected the received data is stored on the disk immediately.

The computed space in the file in case of a loss detection is reserved for several reasons. The first reason is the file system. Most of the existing file systems do not support any efficient insert mechanism, so other mechanisms must be implemented. One conceivable solution would be an index list that contains all the starting points of the packets. With this solution the problem of insertion would be solved, but if a data packet must be searched, a file system seek must performed. As a file system seek consumes plenty of time, it should be avoided. Additionally, either the file system would not behave like a regular file system, or the data would not resemble a regular file.

The solution of reserving the correct amount of space on the hard disk is very simple and efficient, because it preserves the sequential nature of the stored data. And this property is essential for an efficient use of a hard disk, as seeking on a disk importantly diminishes its throughput. Furthermore, this allows LC-RTP to be compatible with multimedia file systems (e.g. [HaSc95], [MNO+94]) which are penalized by inserting or do not support it at all.

Actions after the movie transmission

- SENDER

After sending the BYE message, the sender starts a timer. This timer should be a multiple of the worst case round-trip time (RTT) between the sender and the known receivers. This RTT can be computed with the periodic RTCP packets that are sent for calculations of the network quality. The relevant value can be a worst case RTT, so no special RTT to a special client or server needs to be stored or computed. During this timer period at least one loss list has to be received from a receiver that has detected packet losses. If the timer runs out without reception of such a loss list, the sender assumes that no loss occurred during the transmission and terminates the session completely.

If a loss list arrives, the requested data is stored in a schedule list. This list includes the requested ranges of data and a counter which indicates how many reporting clients miss this specific data range. The counter is incremented if a loss list from a client arrives that includes a request for data that is already included in the sender's loss list. The counter gives an appropriate strategy some information on a schedule for the retransmission of the lost data. A simple strategy might send the data ranges with the highest loss counter at first, because this ensures

that the majority of the cache servers get the lost data early and can then terminate their session and leave the IP multicast group.

Resent packets should be of the same size as the packets that were first sent during the first transmission in order to allow a simple storing mechanism at the receiver's side. The sending mechanism doesn't need to check the range borders but only to check whether the packet has to be stored or not. The byte count that is sent now must be the same as the byte count sent the first time, as otherwise no guarantees of the receiver-sided recognition of the packets can be made. In the same functional procedure as the packet is sent, the schedule list must be updated. This means that the resent data range must be deleted from this list.

When the last entry of the list is processed and deleted, the sender re-sends the end packet in order to inform the receivers that this retransmission cycle is over. The sender repeats now the procedure of setting a timer and waiting for new possible loss lists to arrive. This procedure is repeated until an application-specific retransmission counter has reached its threshold value or until no more loss lists are sent. The retransmission counter prevents the procedure from repeating endlessly in the case of unexpectedly bad network conditions or in case of misbehaving clients.

- RECEIVER

With the reception of the BYE message the receiver finishes the normal procedure of the transmission of the movie and starts the procedure for initiating retransmissions. To avoid a possible overload of the sender, loss lists are sent from the receivers after a random amount of time. This number should be chosen randomly, but below one measured round trip time. The loss list should include all ranges of the detected data losses. If ranges are direct neighbors, they should be combined into one range, in order to keep the size of the list small. This ensures that the additional load of the network remains small. The procedure of sending the loss list after the main movie transmission ensures that no additional network traffic directed toward the end systems arises during the stream transmission of the movie. With this strategy possible network load computations and access control mechanisms need not be changed.

Every retransmitted packet is analyzed to find our whether the byte count in the packet is in the loss list. If it is, the packet is saved at the indicated position in the file by using, if necessary, an offset procedure similar to the one of the sender. Concurrently, the loss list is updated. If the byte count is not included in the loss list the packet is discarded.

When a new end packet arrives, the loss list must be checked. If the list is not empty it has to be sent to the sender again. This procedure is repeated until the loss list is empty, in which case the receiver leaves the multicast group, or until the retransmission counter reaches the application-specific maximum.

To avoid a blocking receiver, the session times out if no end packet or other resent packets are received after a appropriate time, which span several round trip times.

4.6.2 LC-RTP Specification

The design of LC-RTP was made within the constraints of an RFC-conforming RTP implementation. Nevertheless the overview gave a general solution of designing a reliable multicast protocols for VoD-like applications.

The main problem in mapping LC-RTP into RTP is the byte count, as it has to be included into the header of RTP. This is necessary in order to keep content of LC-RTP packages compatible with RTP-related packaging RFCs and therefore to make it possible for standard RTP clients to receive LC-RTP streams. Figure 23 shows an RTP header.

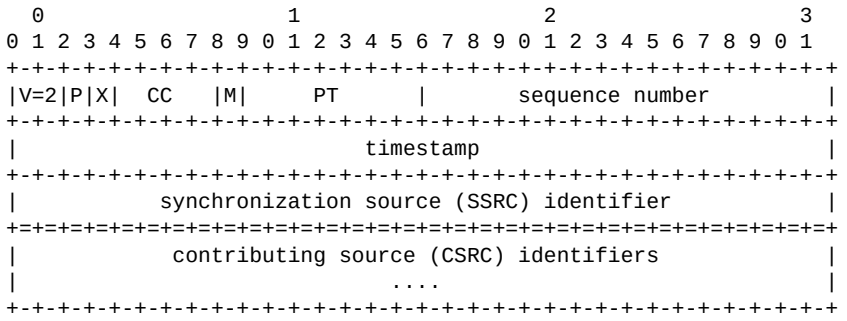


Figure 23: RTP header ([RFC1889])

The only legal way of inserting the byte count into the RTP header and not into the payload is the use of the extension header of RTP (Figure 24). By setting the x-bit a variable-length

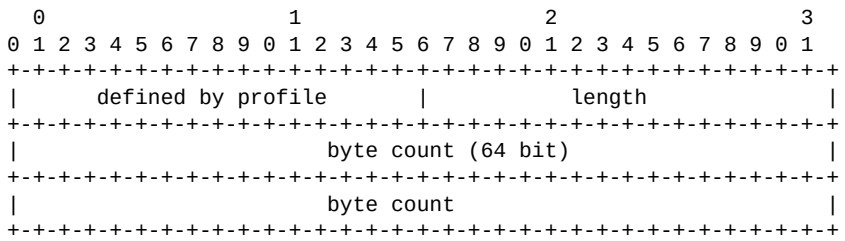


Figure 24: RTP header extension ([RFC1889])

header extension to the RTP header is appended. A header extension contains a 16-bit length field that counts the number of 32-bit words in the extension, excluding the four-octet extension header (therefore zero is a valid length). The other field of the extension header is intended for identifying different header extensions. LC-RTP defines two kinds of header extensions. They are defined to easily distinguish whether a packet is sent as part of the regular stream or during a retransmission phase. The only difference between them is the value in the identifier field. Each extension header has, in addition to the two RTP dependent extension fields, the byte count field. For a current VoD application this field should be 64 bit long, as a wrap around of the byte count must be prevented. For other applications a simple 32 bit word may be sufficient.

4.6.3 LC-RTCP Specification

During the usual movie transmission the RTP transmission is made as usual, except for the byte count which is included in the RTP header. At the end of the transmission an end packet is sent. An appropriate way to do this is by sending an RTCP packet. This packet should not be the normal RTCP BYE packet, as this is used for other meanings. Thus, an application dependent extension RTCP packet must be created. An application defined RTCP packet is shown in figure 25.

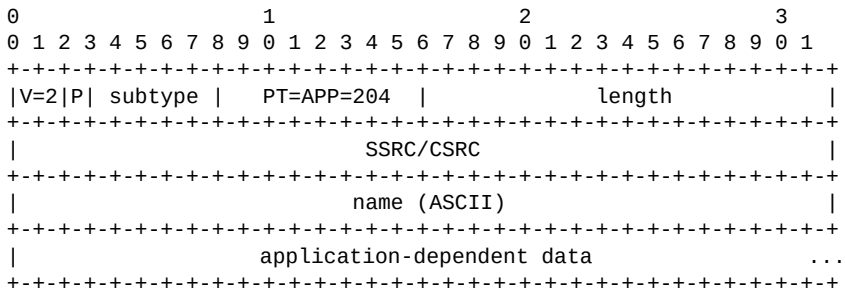


Figure 25: Application defined RTCP packet ([RFC1889])

LC-RTP defines two application defined RTCP packets. The first one is the end packet and the second one is the loss list packet. The NAME field of both packets is set to LRTP, as it has to be a four digit ASCII name.

The only additional data transmitted in the end packet is the last byte count of the session. The name of the packet itself is of enough information for the receiver to interpret this as the end of the normal movie transmission. The list appended into the loss list packet should be appended as a list of byte count ranges. If the loss list exceeds the maximal UDP packet size it should be transmitted in several packets. This avoids any congestion problems with the network.

After the loss lists are sent the sender retransmits the lost data by using the extended RTP packets as shown above. These minimalistic modifications show that the main work of LC-RTP is handled by the logic of the sender and receiver. The extension to RTP is minimal and should be ignored by other applications. In this way LC-RTP is compatible with other applications that participate in the session, like the display tools. This compatibility is very important, because it ensures that a cache server update can be made in parallel to a customer request.

While testing LC-RTP with usual MBone tools an incompatibility was detected. *Vic* and *vat* do not accept any extension to RTP, so they reject all packets with the x-bit set. A comment in the source code explains that an RTP extension is explicitly forbidden through the minimal-control audio and video profile. We have not found any RFC-compliant work-around to this problem, but since *vic* and *vat* implement the variable CSRC list, we have identified at least a non-compliant fix. Since we assume that a cache-based video distribution system would not use mixers, we misuse the CSRC field to transport the byte count instead of the unsupported extension header.

We believe that for the intended application class, the argue that the header extension is sufficiently cheap with an overhead of 8 to 12 bytes per packet. Assuming UDP packets with a typical payload of 512 bytes, our header this causes an overhead of about 1,6%. Furthermore this type of extension is defined in the original RTP RFC ([RFC1889]) and should -theoretically- be implemented by all RTP implementations.

4.6.4 Experimental Results

To investigate the viability of the LC-RTP protocol for wide-area distribution, we have made a set of long-distance unicast transmissions, we have not yet examined the scalability in detail. The long-distance tests are supposed to demonstrate the efficiency of LC-RTP in relation to

TCP, which is a regular file transfer protocol that can retransmit on byte boundaries. In both cases the senders transmit content at most at the regular playback speed of the content.

We have started with the following assumptions:

- In short-distance environments such as LAN and WAN, LC-RTP can hardly compete with TCP. TCP will typically finish the complete transmission earlier because LC-RTP must retransmit a few lost packets while TCP done the retransmissions within the regular playback time.
- In long distance environments, TCP transmissions will usually block during business hours because of congested networks. LC-RTP will experience considerable loss. However, the overall amount of data that is concurrently in transmission will be higher for LC-RTP and may be sufficient to end the transmission earlier than TCP.

The sites that we try to include in our tests are

- KOM at TU Darmstadt
- GMD IPSI in Darmstadt
- ETRI, South Korea
- MCRLab at the University of Ottawa, Canada
- NIST, USA

One of the early results was the experience that LC-RTP is RTP-compliant, but that other RTP tools may be not. E.g. vic and vat, typically the first examples that spring to mind when the MBone and RTP are discussed, crash when they receive streams that include an RTP-compliant header extension.

Our goal for these tests was to show that LC-RTP perfoms as well and reliable as other data distribution protocols (e.g. FTP) and can be used for the reliable distribution of AV content.

We transmited two files one of the size of 6 MB and the other of the size of 20 MB (both MPEG-I movies) from locations in the US and Canada to a receiver located in at our institute. This was performed 5 times for each file from both loacations each time with a different transmission bandwidth.

Bandwidth [kBit/s]	File Size [MByte]	Successful	
125	6	yes	yes
	20	yes	yes
250	6	yes	yes
	20	yes	yes
500	6	yes	yes
	20	yes	yes
1000	6	yes	yes
	20	yes	yes
1500	6	yes	yes
	20	yes	yes

Table 6: Test Results

We decided to perform the tests over a larger distance since we expected to have a higher possibilty of losses than it might be in a LAN or at connections in Germany.

For each test information about the retransmission was logged at the receiver and the original file and the transmitted file were compared to assure that the transmission completed sucessful.

Test results

As shown in Table 6 the two files were always transmitted completly without any errors.The results we obtained from the logging we performed during the LC-RTP sessions show that retransmissions had to be made in allmost all of the test. The logging information also confirmed that the amount of retransmissions increases with the size of the bandwidth we tried to send the files. Which is definetly an expected behavior. If the bandwidth is set much higher than the actual bandwidth of the link between sender and receiver multiple retransmissions for one packet are more likely. But also in these cases the files were transmitted without any errors.

During the tests it also became clear that the quality of the link between Washington, D.C. and Darmstadt is better than between Ottawa and Darmstadt.

Bandwidth [kBit/s]	File Size [MByte]	Max. BW [Bit/s]		Duration [s]	
		NIST	Ottawa	NIST	Ottawa
125	6	1047552	1022800	41	42
	20	1024048	1024000	160	160
250	6	2147480	2045216	20	21
	20	2048104	2048000	80	80
500	6	4294968	3904512	10	11
	20	1561080	4096000	105	40
1000	6	8593216	1169880	5	37
	20	8192008		20	
1500	6	8589936	1213296	5	36
	20	5461336		30	

Table 7: Test Results (Bandwidth, Duration)

We also transmitted both files via FTP from both locations to Darmstadt to obtain some information about the performance of a traditional file transfer protocol.

File Size [MByte]	Max. BW [Bit/s]		Duration [s]	
	NIST	Ottawa	NIST	Ottawa
6	576000	328000	71	126
20	568000	304000	273	512

Table 8: Test Results FTP

Performance of LC-RTP

After the implementation of LC-RTP was finished we did some measurement to confirm the assumption that LC-RTP is on one hand reliable and on the other hand performing at least as well as other transport protocols. Therefore we did some long distance measurement between Germany, the USA and Canada. The test that were performed indicate that both assumptions are fulfilled by LC-RTP.

During the tests we realized that LC-RTP did perform well in point-to-point tests which leads us to the conclusion that LC-RTP must not be used in multicast scenarios only.

Possible Operation Modes

Caching and prefetching of AV content is a powerful method to increase overall performance in the Internet. LC-RTP is designed for this environment. LC-RTP is a simple and efficient reliable multicast protocol compatible with the original RTP. It needs to be implemented only in web servers and proxies. These servers have to be adapted to LC-RTP and they need mainly a list implementation, so the adaptation is a very simple procedure. Other tools are not affected.

All resources are used carefully and the extension permits an implementation to use a simple method to keep the sequential nature of the stored data without buffering. This method considers hard disk performance and possible network structures without wasting resources (like main memory and CPU power). Its intention is to allow a maximum number of concurrent streams handled by the cache servers. As no additional packets are sent during the regular session and the packet sizes are hardly bigger than those of an standard RTP sender, all access control mechanisms and network quality computations can remain unmodified. The only difference to a normal transmission is the fact that after the session, a retransmission of the lost packets to receivers with LC-RTP extensions is performed. A conforming, standard RTP receiver would recognize this as a normal session termination, and would not be affected. Unfortunately, we have observed that popular tools such as *vic* and *vat* do not completely conform to the RTP RFCs. A fix for this situation has been implemented, although LC-RTP's RFC compliance is violated in this case.

By using the same ports as the normal communication, no address conflicts will occur. Multicast ensures a minimum load increase on the network, because the packets are sent only to members of the multicast group, during a transmission to a regular customer.

LC-RTP also supports late joins and early ends of the transmission. The full value of the LC-RTP extension in combination with a special cache server is not yet achieved by simple caching mechanisms. It is necessary to combined this protocol with something like enhanced Patching technique ([HCS98], [GLZS99], [CaLo97]) with LC-RTP, to achieve a relevant decrease in the number of redundant transfers. Since this requires a change in the cache servers' semantics for stream joining (multiple multicast streams must be joint into a single one) we have decided to implement RTP classes with hooks for fine-grained modifications to functional blocks.

4.7 Evaluation

Our protocol suite is designed for this environment to allow the investigation of the most powerful method to increase the system performance of Caching and prefetching in an VoD environment. LC-RTP is a simple and efficient reliable multicast protocol compatible with the original RTP. It needs to be implemented only in library and cache servers, which need an adaptation to the protocol suite. Other tools are not affected.

All resources are used carefully and the extension permits an implementation to use a simple method to keep the sequential nature of the stored data without buffering. This method considers hard disk performance and possible network structures without wasting resources (like main memory and CPU power). Its intention is to allow a maximum number of concurrent streams handled by the cache servers. As no additional packets are sent during the regular movie transmission session and the packet sizes are hardly bigger than those of an non-

enhanced RTP sender, all access control mechanisms and network quality computations can remain unmodified. The only difference to a normal transmission is the fact that after the session, a retransmission of the lost packets to receivers with LC-RTP extensions is performed. A conforming, unmodified RTP receiver would recognize this as a normal session termination, and would not be affected. Unfortunately, we have observed that popular tools such as *vic* and *vat* do not completely conform to the RTP RFCs. A fix for this situation has been implemented, although LC-RTP's RFC compliance is violated in this case.

The protocol suite also supports late joins and early ends of the transmission. However, the full value of the LC-RTP extension in combination with a special cache server is not achieved by simple caching mechanisms. Only in conjunction with a stream scheduling and/or distribution strategies such as those of Chapter 2 and Chapter 3, a relevant decrease in the number of redundant transfers can be achieved. Since this requires a change in the cache servers' semantics for stream joining (multiple multicast streams must be joint into a single one) we have decided to implement RTP classes with hooks for fine-grained modifications to functional blocks.

One further enhancement would be the reduction of the overhead for retransmission, which is a proprietary mechanism and not necessarily RTP-compliant: a combination with the parity-block scheme of [NBT97] for retransmissions may reduce the required bandwidth additionally. It will be even more efficient than the original parity scheme because the LC-RTP is able, based on the loss collection reports, to find the optimal parity group size before starting the retransmission.

A performance enhancement could further be achieve by combining the protocol suite with an adequate file system.It works best with a file system that is designed to reserve space in the file system for the data that would have been contained in lost packets, without relevant performance overhead. This data should be easily inserted into this reserved space when retransmissions arrive at the receiver, again without much performance overhead. Of course, the protocol suite works also with standard file systems (such as Ext2 for Linux) or multimedia file systems (such as IBM TigerShark for AIX). To achieve a better performance, we have been working on a file system called OCFS - the overwrite capable multimedia filesystem, which extends Linux Ext2 with several ideas taken from the Fellini multimedia filesystem. It is not sufficiently developed yet to present details. For a design overview, see Section 10.1.

5. Security and Copyright Protection

The goal of this thesis is the demonstration that decentrally organized wide-area video distribution systems can be a feasible approach towards the achievement of commercial video-on-demand. Therefore, this thesis presents missing links. Historically, video-on-demand systems have been kept closed and centralized, which I consider an inhibiting factor for both the stability of those systems as well as wide-area distribution of multimedia content in general. Commercial success depends on the independance of content providers and service providers; content providers must be enabled to collaborate with an arbitrary number of service providers, as the service providers must be enabled to work with multiple service providers.

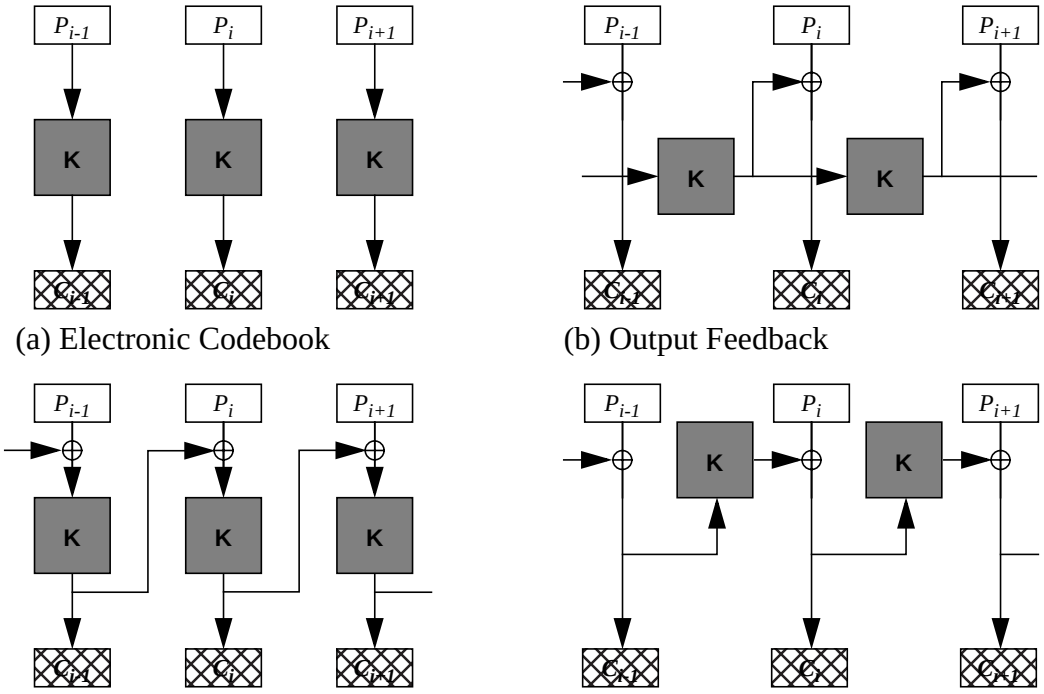
The independance of the collaborators makes such an open environment more susceptible for theft and illegal re-distribution than a closed, centralized system. To address these security issues, this chapter provides an overview over existing as well as new arguments and techniques that make wide-area distribution of video with caching applicable. The aspects that are considered are secure transmission and copyright protection. The first section refers to approaches for encrypted and partially encrypted transmission of video content, which are relevant to the goals of this thesis but which were never a key research topic in preparation of the thesis. This is followed by a presentation of our complementary protection mechanism called Partial Corruption that was initially presented in [GMDS98]. The partial corruption approach requires that consideration is given to protection quality that can be achieved on video encodings, which is done in the following section. After the discussion of protection for the transmission pathes, copyright considerations are addressed in the following sections. First, a general overview of existing watermarking techniques is given, followed by a section that presents a multicast-capable marking approach called the Chameleon stream cipher that was presented in [AnMa97]. After a short section on the error insertion that was applied as an approach for marking in our partial corruption scheme, a final section integrates ideas from partial corruption and Chameleon into a new delivery scheme, Remark, that applies encryption and marking mechanisms for video delivery in distribution chains that include intermediate untrusted storage nodes. With this new approach, it is possible to solve the security concerns in a distribution system with intermediate untrusted caches.

5.1 Secure Transmission by Encryption

Approaches to video encryption can be considered just another application of typical cryptographical tools, and in that case, it is subject to the same attacks or criticisms. Typical encryption approaches can be subdivided into two groups: stream ciphers and block ciphers [Sch96].

Stream ciphers work on single input characters that are transformed based on the key and on state changes in the cipher according to the previous input character. They are suitable for hardware implementation, very fast encryption, or small amounts of data that need to be transmitted with small delay and without bandwidth waste (such as audio data without padding in telephony).

Block ciphers work on larger blocks of input data (typically 64 bits) with transformation based on a key and potentially, on the previous input block. Block ciphers are suitable for



(a) Electronic Codebook (b) Output Feedback

(c) Cipher Block Chaining (d) Cipher Feedback

Figure 26: Operation modi of block ciphers (cf. [Kun98])

- software implementation,
- strong encryption,
- large amounts of data,
- data that is not time-critical and can be collected until an encryption step is affordable, or
- data that can be transmitted with padding bytes.

[Sta95] describes four operation modes for block ciphers: Electronic Codebook, Ouput Feedback, Cipher Block Chaining and Cipher Feedback. Figure 26 shows sketches of the encoding steps of these operation modi, where plaintext blocks P_i are encrypted into cipher block C_i using an encryption step parametrized with a key K. The crossed circle symbolizes an XOR operation. The figure demonstrates the dependance of Cipher Block Chaining and Cipher Feedback on the plaintext. While this characteristic increases the strength of the cipher, its straight-forward application on streamed media requires lossless transmission. Since this can not be guaranteed in our scenario, the application of this kind of operation mode would require session synchronization points. Since Eletronic Codebooks are easily decrypted, this leaves Ouput Feedback as a straightforward mode with streamed video.

The initial approach towards video encryption was the straightforward encryption of the whole stream. Various more efficient encryption algorithms were implemented recently, typically in a way that makes an involvement of an MPEG ([ISO93]) parser necessary. To save bandwidth, partial encryption was introduced. An initial approaches presented in [KVMW98] is based on the content-independent encryption of bytes with constant-sized intervals of unencrypted content in between. While the content becomes unpresentable, the encrypted data is easily identified and can be attacked in a variety of ways.

Maples and Spanos present an approach of partial encryption exclusively of I-frames of MPEG movies ([MS95]). [AgGo96] shows problems in I-frame-only encryption; they can frequently be re-constructed from intra-coded blocks in P- and B-frames. They encrypt P- and B-frames as well. SEC-MPEG ([h:MeGa94]) is an application of this technique - however, the motion vectors do still allow to recognize object borders. The authors of [AgGo96] propose shorter GoPs to increase the number of fully encrypted I-frames, which is inappropriate in our scenario because it increases the required bandwidth significantly.

Tang proposes a scheme of reordered DCT coefficients ([Tan96]). This has the advantage of operating on decompressed video with an overhead that is close to negligible since only the order of DCT coefficient processing must be changed. Futhermore, the protection against brute force attacks is better than in standard encryption approaches such as DES and IDEA. However, the efficiency of compression is reduced, which increases the required bandwidth; statistical analysis allows frequently a re-ordering of the DCT coefficients. Another criticism is that the approach requires strong interaction with the presentation software and hardware; inserting this encryption on demand is extremely expensive since the content needs to be fully parsed.

Qiao et.al. propose a scheme called VEA (video encryption algorithm) that works exclusively on the data bytes and does not interpret the MPEG data ([QNT97]). They exploit the entropy in the MPEG data stream to use part of the video itself as a one-time key for the video. The video is segmented into blocks that have a size that fits to the underlying encryption algorithm. Every other block is XOR'ed with its neighbour and left otherwise unmodified. The neighbour block is fully encrypted. This approach reduces the cost of encryption by about 47%. Still, each byte of the video data is manipulated once for each transmission. It is immanently important to use this approach in a content-independent way because an analysis could otherwise decode the XOR'd elements and use this knowledge to find the encryption key. The probability of a succesful attack grows when the approach is applied recursively.

Kunkelmann ([Kun98]) works on JPEG-based codecs and requires a modified parser for partial encryption. Only the DC and low frequency AC coefficients are encyrpted. The approach is flexibly scalable by the number of encrypted coefficients. The most relevant initial coefficients of a DCT are always encrypted, and depending on a parameter that defines the strength of the encryption, additional coefficients of the DCT block are encrypted as well. A refinement makes different decisions for inter- and intra-coded blocks. Kunkelmann et.al. present in [KRSB97] a variety of applications of this partial encryption of the complete video stream, for use with a security gateway, and come to the conclusion that a mix of partial bit stream encryption and variable length code encryption is the most efficient for their application. They consider a partial encryption of 10% of the data appropriate for VoD applications, while full protection requires a major part of all data to be encrypted to prevent reconstruction.

5.2 Secure Transmission by Partial Corruption

Under the impression of the video protection by partial encryption, we developed another fast and computationally cheap solution and presented it in [GMDS98]. In contrast to typical approaches to video protection by encryption, which are generically applicable, including applications such as video conferencing or video archiving, our goal was specifically the protection of videos in a wide-area delivery system in which (potentially encrypted) videos are

stored on untrusted and (potentially) insecure intermediate nodes for a relatively long time. The specific goal of commercial video-on-demand inspires two demands:

- **futility**: a theft of movies that are stored in untrusted caches should be futile
- **notification**: it should be necessary to contact the content provider for each retrieval of a movie

The existing encryption approaches are relatively computing intensive and put a heavy strain on a VoD server when they are executed in real-time. Since an encryption key can not be re-used for any two receivers due to the *notification* demand, tranditional encryption mechanisms have to be performed for each receiver of a stream independently. Kunkelmann et.al. report an increase of CPU utilization by 10.5% for the playback of the video stream when decryption is necessary. The server would spend this amount of computing power per delivered stream. Even if the computing power of servers would increase in such a way that these resources could be expended, such a scenario has additional negative impact on the tuning options that are available for servers. Since large parts of all data streams need to be manipulated for each request, the optimizations which have been investigated in earlier video server work and work by aggregating requests such as batching ([DSS96]) are reduced to schemes for unloading the servers' disks; memory, CPU and networking requirements grow linearly with the number of concurrent requests due to the per-usage processing.

Our intention was the development of a mechanism that is able to work with caching and prefeteching to reduce the requirements of the networking infrastructure for our distribution system. The mechanism should be computationally cheap for the content provider's servers and the cache servers, in order not to overload the server with the task of modifying the content for an arbitrary number of concurrent unicast transmissions which would make the application unscalable.

With the two goals in mind, the Partial Corruption approach was developed, which addresses these demands but does not provide a generic solution to cover video encryption in general. For some multimedia applications such as video conferencing, perfect protection is a major requirement of the communicating partners. In commercial VoD, not the perfect protection of content is a necessity, but a reduction of the viewing quality below an acceptable level is already sufficient to solve the *futility* demand.

Partial Corruption works by unencrypted, but partially corrupted transmission of the bulk of a movie's data, which allows the use of caching, and the additional encrypted point-to-point transmission of a minimal amount of data that is necessary to reconstruct the complete multimedia content. This unloads the servers and networks from the necessity of client-specific encryption and from the subsequent point-to-point transmission of complete movies; it allows the use of caching for the corrupted, major part of the movie. The corruption works by destroying part of the data entirely in the freely distributed part of the movie. Since the data that is required to complete the movie is not present in the corrupted movie at all, attackers can not break any encryption algorithm to decode the missing information.

In contrast to many partial encryption approaches, the corruption of movie data in our approach is content-independent. Content-independance, in this context, means that for chosing parts of the original movie that are to be corrupted, no consideration is given to its content,

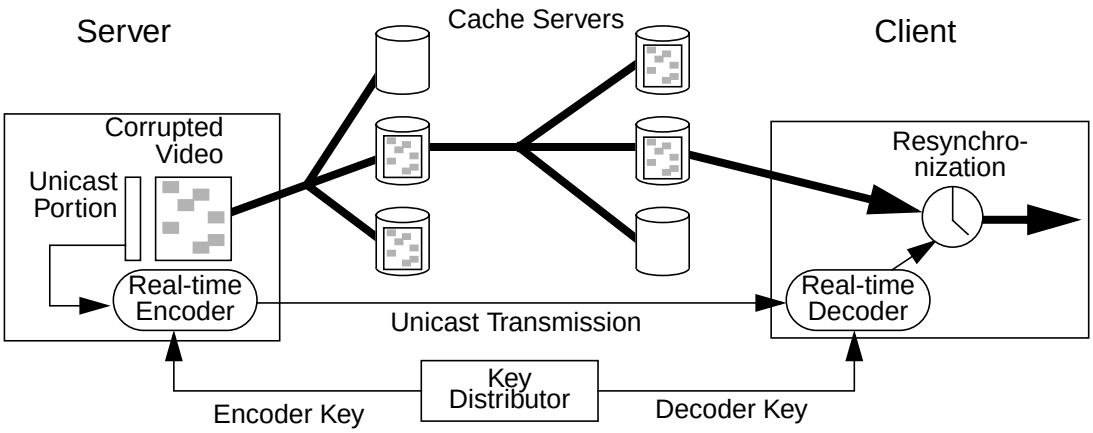


Figure 27: Distribution System for the Partial Corruption approach

such as video data, audio data, or header information. Correct replacements for the corrupted part of the data are transmitted to the customer by means of an encrypted point-to-point connection.

The organization of a distribution system that supports this protection mechanism is sketched in Figure 27. The video transmission is performed in two phases. The bigger part of the video is corrupted and it is made (from the content provider's point of view) publically available in cache servers in the first phase (this part is labelled "corrupted video"). In the second phase, a point-to-point transmission is used to deliver the missing bytes to the customer. These missing bytes are fully encrypted with a client-specific key when a video is actually requested ("unicast portion"). This scheme provides the content owner with the information that a request has taken place, which solves also the the *notification* demand. It gives the content provider the billing option, and provides an identification of the receiver of a perfect copy of the movie.

The unicast portion is encrypted on the server side using a personal key of the receiver, e.g. a key provided by a trusted third party. If the unicast portion is small in comparison to the complete video, the computational load of encrypting this portion of the video is relevantly below that which is induced by using an MPEG parser. Also, less interaction with the optimized output paths of video servers or video cache servers is necessary.

At the receiver's side, the unicast portion is decrypted using the personal decryption key, and established synchronization approaches (e.g. from [JG97]) can be applied to synchronize the unicasted partial transmission with the main part of the data which is received from a nearby cache server.

The decision whether the encrypted data can be consumed directly from the data stream or whether a download is necessary depends mainly on the observed throughput. It can be made independently by the client while the stream is being received. Assuming that the unicast portion makes up 1% of an MPEG-1 video and repairs defects that are uniformly distributed over the length of the movie, the required point-to-point throughput is approximately 2 KB/s, which is streamable in large parts of the Internet nowadays. For the bulk of the video data we assume the presence of a cache server to which a 1.5 MBit/s streaming connection can be established.

5.3 Protection Features of Partial Corruption

To the extent described in the previous section, our approach can be applied content independently and format-independently on all kinds of streamed media data. This does not mean that the approach provides an effective protection for all encoding formats. The small subset of data that we intend to corrupt is not generally sufficient to confuse an arbitrary encoding algorithm. The reason for this is not the generic use of encoding formats but the reduction of the knowledge about the stream which is necessary to corrupt the data sufficiently to remove perceptible information. We have applied the scheme to MPEG-1 [ISO93], for which it is appropriate. It is certainly also applicable for MPEG-2 [ISO96] or H.263 [ITU96] with Huffman coding [Huf52]. Other video compression approaches such as fractal compression should be affected as well. For plainly intra-coded formats such as Motion-JPEG [ISO93b], reconstruction by comparison of neighbour frames can be automatized easily; applying our scheme to this format requires at least a reconsideration of the content independence.

In typical MPEG-1 groups of pictures (GoPs), however, I-frames, which are the basis for repairing frames, are sufficiently far apart in time to make this automatic reconstruction difficult for large parts of the video. D'Ardia et. al. have presented results in [DFV97] that show a relevant variation between consecutive I-frames in most kinds of video transmission such as sports, news or movies. The scheme could probably not by applied to talk-shows because the low variation between consecutive I-frames allows reconstruction of the sequence.

We have investigated the percentage of data that needs to be corrupted in an MPEG-1 video stream to reduce the video quality to a 'teaser' or worse quality. 'Teaser' quality is a presentation quality that makes a content easily recognizable but unenjoyable without decryption, repair or descrambling. PayTV networks are frequently applying 'teaser' quality scrambling to attract customers. In contrast to other approaches, which work on the uncompressed images, we make use of the two vulnerabilities of MPEG to data corruption or data loss ([GMDS98]):

- error propagation by relative encoding through motion vectors
- error propagation by decompression

The first vulnerability is that the destruction of an MPEG-1 I-frame affects all frames in the following GoP. In a video that has been compressed with a typical GoP length of 15 frames, the error is expanded in time by the relative decoding in P- and B-frames and will often affect all 15 frames.

The second vulnerability is introduced by the compression scheme. MPEG-1's Huffmann compression improves the effectiveness of our intentional corruption of single bytes of data. Since the Huffmann algorithm is bit-oriented rather than byte-oriented, a typical Huffmann decoder implementation is unable to recover from the error for the rest of a data segment. Furthermore, a complete Huffmann decoding of the data is necessary before the corruption is detected because all bytes except for the special values 0xff and 0x0 are meaningful to the Huffmann decoder. As a result of this error propagation from a corrupted byte to the rest of a data segment in a frame, the number of bytes that need to be destroyed to corrupt a compressed MPEG frame completely is much lower than the number of bytes necessary for an uncompressed frame. To verify this second vulnerability, we have tested various clips and parameter sets. Tested players are listed in Table 9.

All of our test clips were approx. 1.5 MBit/s MPEG system streams from various sources and decoders. Movie lengths were between 17 and 615 MBytes. We have performed the playback experiments using various MPEG-1 players on various platforms:	
Berkeley mpeg_play	SUN Solaris
MpegTV's mtv	Linux is a trademark of Linus Torvalds
MpegTV's mtv	SUN Solaris
IBM VideoCharger Player (based on Microsoft ActiveMovie)	Microsoft Windows 95
IBM VideoCharger Player	Microsoft Windows NT
Since mpeg_play is incapable of handling errors, and both mtv and VideoCharger work more or less identical on all platforms, these are effectively two tests.	

Table 9: Players used for Partial Corruption Experiments

Because of the error propagation, the destruction of larger blocks with the same overall ratio of corrupt to correct bytes turns out not to be appropiate. The reason for this is the effect that Huffmann decoders generate corrupted data from the bytes immediately after the first corrupted bit. This effect is not increased by longer series of corrupted bytes. The corruption of single bits may be as efficient as the corruption of bytes, but it is not advisable in our case because the bit changes increase rather than decrease the number of CPU operations.

As mentioned above, the mechanism of our approach is content-insensitive. It is generally assumed that reconstruction of headers for MPEG-1 is relatively simple because current encoders produce CBR streams and use always the same header data anyway. Thus, in all our experiments we took care that all headers of a video are correctly re-inserted. The remaining errors are disturbing enough to yield results that are unacceptable for commercial exploitation. We have also reconstructed the audio parts of the stream in most of the tests for a few reasons. First and foremost, the players synchronize their timing to the audio clock and we wanted to remove the effect of timing errors from our experiments. Second, in spite of the continued operation of the ActiveMovie player, Windows would frequently hang when the audio device drivers received too much corrupted data. Third, measuring the level of destruction in an audio track requires devices that we did not have available.

Quantitative Observations

We started experiments with a destruction ratio of 1%, assuming that this would not be sufficient to destroy the video sufficiently to render it unviewable. However, using an error size of 1 byte, the error propagation rendered movies unplayable to two MPEG players (ActiveMovie, VideoCharger Player) and showed only artifacts in other (MpegTV). This observation was made even after full reconstruction of all headers and the complete audio stream. Figure 28 demonstrates the effect that the corruption of 1% of randomly chosen bytes has on a JPEG image. Since the basic compression mechanism of JPEG images and MPEG I-frames is the

Figure 28: Original JPEG image and the same image after the corruption of randomly chosen 1% of all bytes

same, the reader gets a good impression of the effects that this destruction has on an MPEG movie.

Subsequently, we considered even lower error rates and found 0.5% to deliver bad quality and 0.1% to provide a quality sufficient to read blocks of text that remain immovable for several seconds, e.g. in the trailers. With these error rates we are still adequately above current capabilities of restoration to good quality. Currently, these mechanisms are able to handle bit error rates of about to 10^{-7} well (which is equivaluent to a lower byte error rate due to possible multiple hits in bytes). The rather limited restoration capabilities may be due to the fact that this kind of restoration is not a research topic at this time.

Here, a question comes in to focus: which options are available to a data pirate? Most probably, we have to add additional security mechanisms. We assume that the encryption algorithm and the key exchange mechanisms protect the data from being stolen by an eavesdropper during transmission (Figure 27) and thus, that the encrypted part of the content remains safe. The primary concern is then whether the unencrypted data is protected from restoration.

In our experiments we have distinguished *fixed* and *variable* byte values used for the corruption of the original stream, and the applications of this corruption at *periodic* or *variable* offsets from each other.

An attacker may easily identify both a fixed byte value (by gathering statistics on frequencies, see Figure 29) and a periodic offset (by the use of auto-correlation). Since this makes restoration easier, neither should be used to ahieve better concealment. To prevent the identification it is essential to vary both the replacement distance and the replacement value.

Achieving variable offsets

To get around the corruption of bytes at equal distances throughout the video, we apply the Poisson distribution and a random seed per movie. We write bytes from the original video to a file (the unicast portion) and afterwards, destroy those bytes in the original video. The seed value is transmitted to the receiver before all other content that is sent in the unicast transmission. The client's implementation of the distribution function must be identical to the server's,

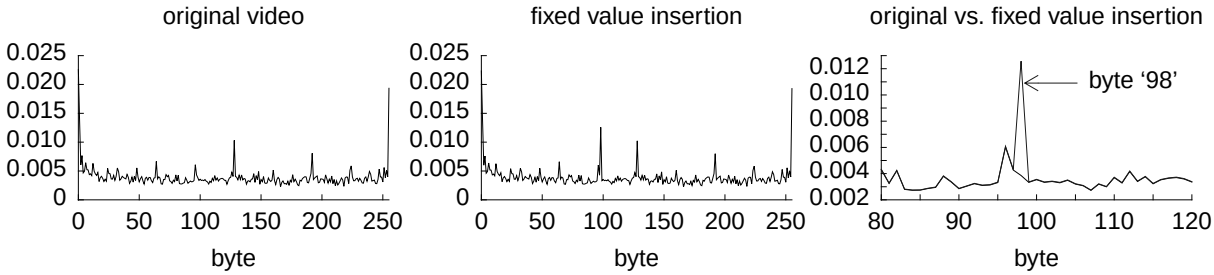


Figure 29: Byte value frequencies in original movie and after fixed corruption

e.g. we have used the rather limited, but floating-point free, implementation of the CNCL library [h:CNCL96] with 32 bit signed integers. Under this assumption, the client is able to replace the corrupted bytes with the decrypted bytes when it receives them over the unicast connection.

Selecting appropriate replacement values

In our first approaches we evaluated the effect of inserting constant values (bytes) into video streams for simplicity reasons. However, we found that such bytes were too easily identified by an attacker. We have responded to this by trying to replace the correct byte with a corrupt byte in such a way that the chances of identifying this byte as corrupt by statistical analysis of the stream or part of the stream is minimal.

To do this, we have examined the entropy of the video data. Signal theory states that a high information density of a data set requires a high entropy value of that data set. The entropy is computed as follows: the relevance *I* of a byte *i* value depends on the frequency h_i of that byte *i* in a data set ([Hil87]).

$$I(h_i) = K \cdot \log h_i \qquad ;K = -\frac{1}{\log 256} \qquad (1)$$

With (1), the entropy *H* is calculate as the average relevance of all byte values by

$$H = \bar{I} = \sum_{i=1}^{n} h_i \cdot I(h_i) = K \cdot \sum_{i=1}^{n} h_i \cdot \log h_i \qquad ;K = -\frac{1}{\log 256} \qquad (2)$$

Formula (2) allows the computation of an information difference $\Delta I = I(h_{new}) - I(h_{orig})$ between a data set h_{orig} and a modified data set h_{new}. A positive value indicates and increase in entropy, a negative value indicates a decrease in entropy. If the entropy value is high all for all subsets of a data set, it appears chaotic.

We have examined the entropy value of MPEG-1 movies and we have found high entropy values between 97.4% - 99% in our example movies, indicating a high information density - which is also perceived as a high level of randomness or a as very chaotic appearance. If the goal is to present a less informative stream you have to present something like a blank paper. The optimum would be that some special chosen bytes are presented very often and the frequency of the other values converge to 0. The probability of finding less frequent values is low but especially those values should be changed. However, the most frequent bytes in MPEG are

header and padding bytes. Thus, a byte that assumes either of these values that is obviously in the wrong place could be identified quickly, which simplifies reconstruction of the original (Figure 29 demonstrates the ease of a statistical identification of such a value). We concluded that lowering the entropy in this way is ineffective. To hide the corrupted bytes in such a movie, it is appropriate to use an approach of “desinformation”: we try to select corrupted bytes for insertion in such a way that the entropy is strictly increasing. At the same time, this corrupted byte replaces a correct one in a data set of high information density, which implies that a byte with high information content is removed - the impact of the replacement on the information itself is extremely high. Since the increasing entropy is perceived as an increase in chaotic appearance of the content, it is harder for an attacker to indentify the corrupted byte by statistical means.

Implementation

Although not strictly required by the Partial Corruption approach, we apply our corruption approach to the data of a video stream as it is streamed. We are able to control whether we want a higher or a lower entropy because we can choose the value of the corrupted byte in the output stream freely.

Control or padding bytes have high frequencies. Thus, their relevance in the calculation of the entropy value *H* is low according to formula (2).

Because the entropy of compressed video streams is typically very high from the start, it is most effective to choose the least frequent byte from the stream to replace the original value at any position. To increase entropy means that all values should come close to an identical frequency.

Thus, choosing a value with high frequency (and it is very probable to be hit) and replacing it by a value with low frequency (optimum would be the value with lowest frequency according to formula (2)) has the greatest effect on the overall entropy value, which reaches its maximum with a uniform distribution of all byte values.

For our scheme, we have decided to simplify the finding of these least frequent values. Instead of collecting statistics of the complete video stream, we select the value for insertion by identifying the byte *i* with the lowest relevance $I(h_i)$ by calculating $I(h_i) = k \cdot \log h_i$ on all bytes that have been observed previously in the stream.

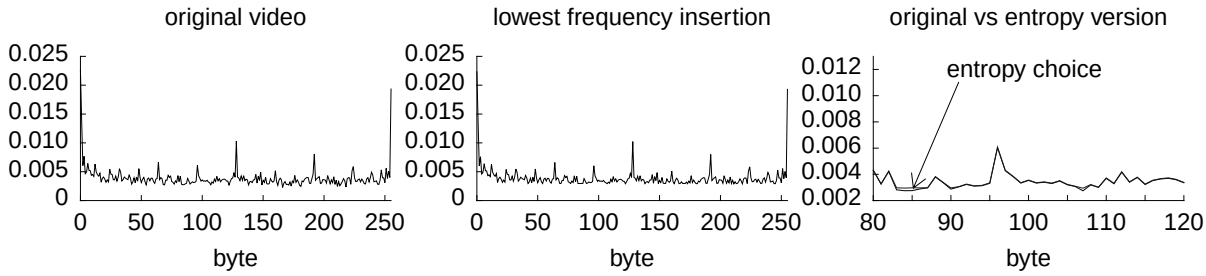


Figure 30: Byte value frequencies in original movie and after statistical corruption

Figure 30 demonstrates the nearly invisible changes in byte frequencies that can be achieved by this means: even the differences in the byte frequencies between original and manipulated

movie are hardly detectable. Thus we assume that the optimal fill byte value for hiding the replacement inside the stream can be found by observing the byte frequencies. Note that the probability to change a byte value with high frequency is high, although not optimal in our case since we do not search actively for frequent bytes in the stream.

A potential danger to this approach would be the use of exactly the same formula in an attack to identify the positions that have been manipulated by us. Since the entropy of MPEG streams is extremely high from the start, this approach can not be used to identify the positions of these bytes once that they have been inserted following our approach.

Conclusively, we can state that this approach to partial corruption is a direct counterattack to frequency analysis that is used in all statistical attacks.

5.4 Watermarking for Copyright Violator Tracing

The schemes presented so far protect from the theft of data that is moved into and stored in caches. However, an authorized receiver of the movie, who has the full quality data available, may choose to record and resell it. Such a customer is typically considered a *copyright violator* (sometimes also called a *traitor*).

The prevention of such copyright violations is hardly possible unless dedicated hardware is used that is protected from manipulation by the customer. Since such an approach is unlikely to succeed with video-on-demand (consider analog capturing of television screen), the proof of copyright violations is a more realistic issue. Major efforts in this direction are currently conducted by watermark researchers under the label *fingerprinting*. The idea is to insert customer-specific marks into a movie. The unauthorized reseller may decide to request the movie multiple times or to cooperate with another copyright violator in order to use a collusion attack (a voting mechanism) to eliminate the marks (we must always assume that the protection technique is known). It is important to find a scheme that will yield a sufficiently large number of remaining marks to single out the unauthorized reseller and take further measures to prove the contract violation. In this context, it is acceptable that the content provider needs to use a brute force approach to identify marks that remain after the execution of voting steps. Fingerprinting is basically watermarking with the specific goal of identifying copyright violators, while watermarking is more generally applicable. Thus, an introduction requires an explanation of watermarking and its application in fingerprinting.

Watermarking is applicable to all kinds of media: still images, audio, video, and even 3D modelling. In this context, mainly video is of interest. Many of the existing video watermarking scheme are extensions of still image watermarking techniques. The latter must perform well with respect to the following criteria ([Ditt99]):

- visibility: often, the watermark should be hidden completely from the human perception; in rare cases, it is supposed to be visible
- robustness: a watermark should be resistant to attacks such as de- and re-compression, geometrical transformation, shearing or zooming
- capacity: the amount of information that a watermark can carry
- complexity: the introduction of a watermark into a content should be of limited complexity
- security: the ability to exactly identify one or more copyright violators

Video adds changes the requirements to a watermark and adds some more criteria to decide whether a marking approach is applicable ([Ditt99]):

- authenticity of order: the watermark should be resistant to, and possible prove, a change of order in the frames of a video
- modified visibility requirements: watermarks that remain invisible in still images may become disturbing in videos
- modified robustness requirements: watermarks need to resists cuts, re-ordering, interframe-coding, filtering and scaling
- modified complexity requirements: the performance of insertion and detection algorithms is even more important for long videos
- modified capacity requirements: the insertion of information into videos simpler than in the case of images since more data is available to insert the marking information
- error correction: the increase in available capacity allows the introduction of redundant information into the watermark

Fingerprinting consists of four elements:

- the identification of potential markable spots in the content that must be protected
- a watermarking approach to introduce the information into the content
- a fingerprinting algorithm that selects the marks for insertion, based on the number of receivers and the required security level
- a detection tool that can identify the copyright violators who have cooperated to remove the marks in their respective copies of the content

[Ditt99] provides a very recent overview of existing watermarking and fingerprinting approaches as well as new approach and detailled evaluations.

According to [DBS+99], a fingerprint that is intended for unqualified identification of copyright violators who cooperate to identify and remove the unique marks that they have received must fulfil the following condition: if a protection against a collaboration of up to d attackers (a so-called collusion attack) from a set of up to q users is requested, it is necessary to identify at least $\sum_{i=1}^{d} q^i$ marking positions in the movie. The number of marking positions is identical to the length of the fingerprinting key. Keys are generated in such a way that each group of d customers from the q potential customers share exactly one mark that is not shared with any other customer. This computation implies that the attackers have an advantage over the defenders of the copyrights. For a linear growth in the number of collusion attackers, the size of fingerprint must grown potentially.

These marks must be hidden in the movie by a watermarking approach that is unperceivable to humans; it is also important that the same marks from different keys are introduced in such a way that a collusion attack can not identify the existance of the mark from the difference of their marking effect. [Ditt99] presents a watermarking approach that is able to introduce such fingerprints with additional redundancy to increase the robustness of the mark.

All existing fingerprinting schemes have a relevant drawback: either the marks are introduced at the receiver side of a transmission, which is inherently insecure, or they require the personal delivery of the marked content to each customer and thus, can not interoperate with multicasting or caching.

An alternative technique that can be used to insert personal marks, and which can be extended to use multicast and caching is the insertion of random marks. Like a regular watermarking approach, this can be exploited to prove copyright violation in a way that makes the danger of manipulation to the decoder software irrelevant. The following sections are concerned with such techniques.

5.5 Marking with the Chameleon Stream Cipher

Chameleon is a stream cipher for the integration of copyright violation tracing for data streams that are distributed by multicast. It was proposed in [AnMa97][1]. Roughly, it works by encrypting a data stream with one key, by transmitting the encrypted stream and then inserting pseudo-random errors into the data streams during decryption by only providing decryption keys that are slightly different from the encryption key (an application-dependent number of bits is negated). The generated error is minimal and differs from one customer to another. If the content can be segmented into equally-sized blocks, such as uncompressed audio streams, the scheme allows the exclusion of errors from large parts of these blocks.

Specifically, Chameleon is not assumed to be working on its own, but always in combination with a stream cipher that does relevantly stronger encryption than Chameleon itself - Chameleons goal is primarily the introduction of marks into the stream, not encryption. The original Chameleon mechanism works as shown in Figure 31.

1. First, a key a for a random number generator is chosen.
2. Next, a set of random long words is chosen, e.g. 512 kByte of 64 bit words, ie. 2^{16} words. This is called the key B.
3. Using the random number generator (initialized with key A), k 64 bit-aligned 64 bit words are chosen from key B.
4. The k chosen words are XOR'd with each other and with a word of the plaintext.

Figure 31: The Chameleon algorithm

We extend this mechanism by random permutation of the chosen long words of step 2. Without that additional step, marks would always appear in exactly the same bit position, and collusion

1. Thanks to Fabien Peticolas, who pointed to this protection approach that works with multiple decryption keys.

attacks have a better success probability when the number of error bits in the key B is low. The extended mechanism works as shown in Figure 31.

1. First, a key a for a random number generator is chosen.
2. Next, a set of random long words is chosen, e.g. 512 kByte of 64 bit words, ie. 2^{16} words. This is called the key B.
3. the words that are chosen in step 2. are permutated and shifted randomly (by drawing 64 bit words from the keystream generator for XOR'ing and shifting), which is be an extension to the original scheme
4. Using the random number generator (initialized with key A), k 64 bit-aligned 64 bit words are chosen from key B.
5. The k chosen words are XOR'd with each other and with a word of the plaintext.

Figure 31: The extended Chameleon algorithm

Using this scheme, the stream is encrypted and transmitted to the customer. Each customer receives a personal key B', which differs from B in a variety of random bits, and receives the key for the random number generator A. Computations in Section 10.2 indicate that 2^{12} is a good number. The customer owns also the random number generator. Using key A to start the random number generator, the customer draws the exact sequence of words from B' as they were used for the encryption, and permutates key B' in the same way as well. The resulting k words are XOR'd with the encrypted content and produce the defective plaintext. With the number of XOR'd words k=4 and $4096=2^{12}$ marked bits, this bit error rate reaches just below 0.1%.

Problems

The Chameleon cipher is a lot cheaper in terms of data transfer than the partial corruption approach, since e.g. it does not add any transmission overhead to the original stream at all. But one of Chameleon's problems is that the movie is still perfectly intact until it is decrypted, which allows an attacker who is able to perform a decryption to escape unrecognized.

In spite of the good value for the number of bit errors in a key, this introduces another problem which becomes visible when the stream is decrypted with the personal B' key. With this number of marked bits in keys, and a straight-forward use of this key in decryption, we have an unacceptable number of errors in the decrypted movie. For example with approach applied to an MPEG-1 codec, this would be ruinous as the bit error rate effects a .75% byte error rate which we have shown to be sufficiently destructive to distribute the remaining information entirely without encryption (see page 81).

The Chameleon paper offers an approach to this problem. If too many errors are counterproductive, use key A to select bytes that can be defective; Figure 32 shows such a selection. Since A is unique per movie and not per copy, all copies in a collusion attack are affected in the same bytes. These bytes will be identifiable by the attackers because they have the keystream generator available but the protection from collusion attacks remains. The reason for this is not explained in the original paper, but the reason that an attacker would have problems to replace a manipulated byte with a correct one is the high entropy of the movie (compare with the error

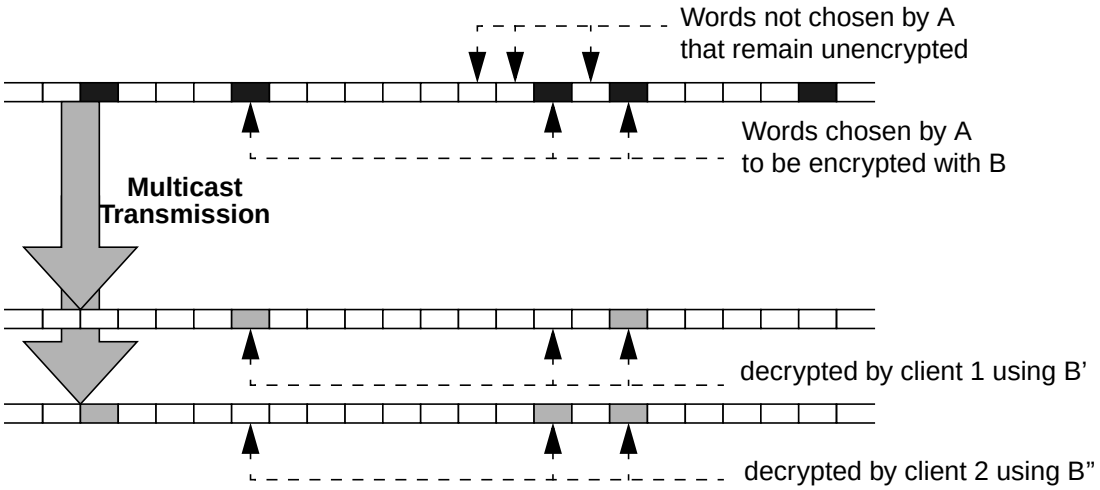


Figure 32: Partial application of Chameleon for movies

byte hiding on page 83). The brute force attack could work by using an output error detection to identify errors in the display and to modify the original byte accordingly until the error is repaired.

Another problem arises from its modification of bits which is, in contrast to the partial corruption approach, mainly uncontrolled, i.e. with Chameleon, it is impossible to prevent the destruction of header information when a compressed VBR (variable bit rate) stream is destroyed. This header destruction can only be prevented by working on the uncompressed video data stream. In a VoD scenario, there is also arbitrary time and computing power available for off-line creating of the encrypted stream at the server side. This causes a problem at the receiver side, where the decompression and display facilities for the video streams are typically implemented in hardware (MPEG-1 is implemented in hardware in dedicated devices, MPEG-2 uses hardware decoding in nearly all devices); the transport and demultiplexing facilities, on the other hand, are typically seperated from the decoding because additional information needs to be extracted, e.g. MHEG objects in the recent UK DVB installation. Thus, it would be more convenient to have a mechanism that marks compressed data.

The Chameleon paper talks about audio data, which is typically decompressed completely before being passed to the hardware. However, there is no efficient and reliable audio watermarking scheme available yet ([Ditt99], p.76).

Note that the random number generator, or keystream generator, can certainly not accept feedback from the operations on the B key. Operations on the content are permissible only under the (usually invalid) assumption that there is no packet loss in the video transmission. If there is rare packet loss, key B feedback can be applied under the condition that the keystream generator implements session synchronization points, which reduces the efficiency of the feedback partially.

5.6 Protection Features of Chameleon

The interesting feature of the Chameleon approach is that it protects both the key B, which is easily available to the receiver, and the content. A collusion attack on the key itself will fail

with a high probability; the remaining bit errors allow to reliably identify the attacker. A proposal is to use a key length of 2^{22} bits, and to insert about 2^{12} error bits into the key for each receiver. The probability that a bit error is identical in the keys of two attackers defines the probability that an error is not identified by two collaborating attackers. With a key length of $N = 2^{22}$ bits and a total of $n = 2^{12}$ (resp. $n = 2^{11}$) bit errors per personal key there is a probability the 2 attackers fail to identify at least one bit of $P_{12} = 0.982$ (resp. $P_{11} = 0.632$). This is identifies a set of possible keys and attackers, which is rather small since the chance for a person to have this error bit is 2^{-10} (resp 2^{-11})(computation in Section 10.2.1).

With a 3-party collusion attack, this is a bit more complex; a collusion attack may fail in two cases. Either a bit error may remain unnoticed, ie. all 3 keys have an error bit in common, or a vote may be incorrect because two keys have an error bit in common. With a 3 party collusion attack and a key length of $n = 2^{11}$ bits, the probability of any remaining errors is $P_{11} = 0.314$, with $n = 2^{12}$, the probability is $P_{12} = 0.963$, and with $n = 2^{13}$, the probability that at least one bit remains unnoticed is $P_{13} \approx 1$ (computation in Section 10.2.2). In case of the same error density n/N but a longer key, the probability that a collusion attack on the key itself fails is increasing. For example, $N = 2^{23}$, $n = 2^{12}$ yield an error probability of $P_{12} = 0.73$.

The decryption of the content introduces the bit errors of the keys into the presented content. Since each bit of the key is applied randomly as one in a group of four bits, and the operation that is performed is an XOR operation on for randomly chosen words, the bit error probability is

$$P(\text{bit in decrypted content is wrong}) = \frac{4 \cdot \left(n \cdot \frac{(N-n)!}{(N-n-3)!} + (N-n) \cdot \frac{n!}{(n-3)!} \right)}{\frac{N!}{(N-4)!}}$$

which yields error probabilities and average error numbers as presented in Table 10.

key length error bits	$N = 2^{22}$ $n = 2^{10}$	$N = 2^{22}$ $n = 2^{11}$	$N = 2^{22}$ $n = 2^{12}$	$N = 2^{22}$ $n = 2^{13}$	$N = 2^{23}$ $n = 2^{10}$	$N = 2^{23}$ $n = 2^{11}$	$N = 2^{23}$ $n = 2^{12}$	$N = 2^{23}$ $n = 2^{13}$
error prob.	0.00098	0.00049	0.00194	0.00098	0.00389	0.00195	0.00775	0.00389
errors/MB	8184	16352	32640	65025	4094	8184	16352	32640

Table 10: Error probabilities for different key lengths

Since the error distribution from the modified key into the decrypted content is linear, it is appropriate to compute the probability that a collusion attack identifies all marks in a movie from the amount of marked data, the key length and the detection probability for that key length.

5.7 Marking Extensions in Partial Corruption

We have examined a couple of schemes that insert infrequent byte errors into the video stream randomly and found that completely random errors are easily fixed by applying voting mechanisms. The initial idea that we presented in [GMDS98] was to choose a random sequence of

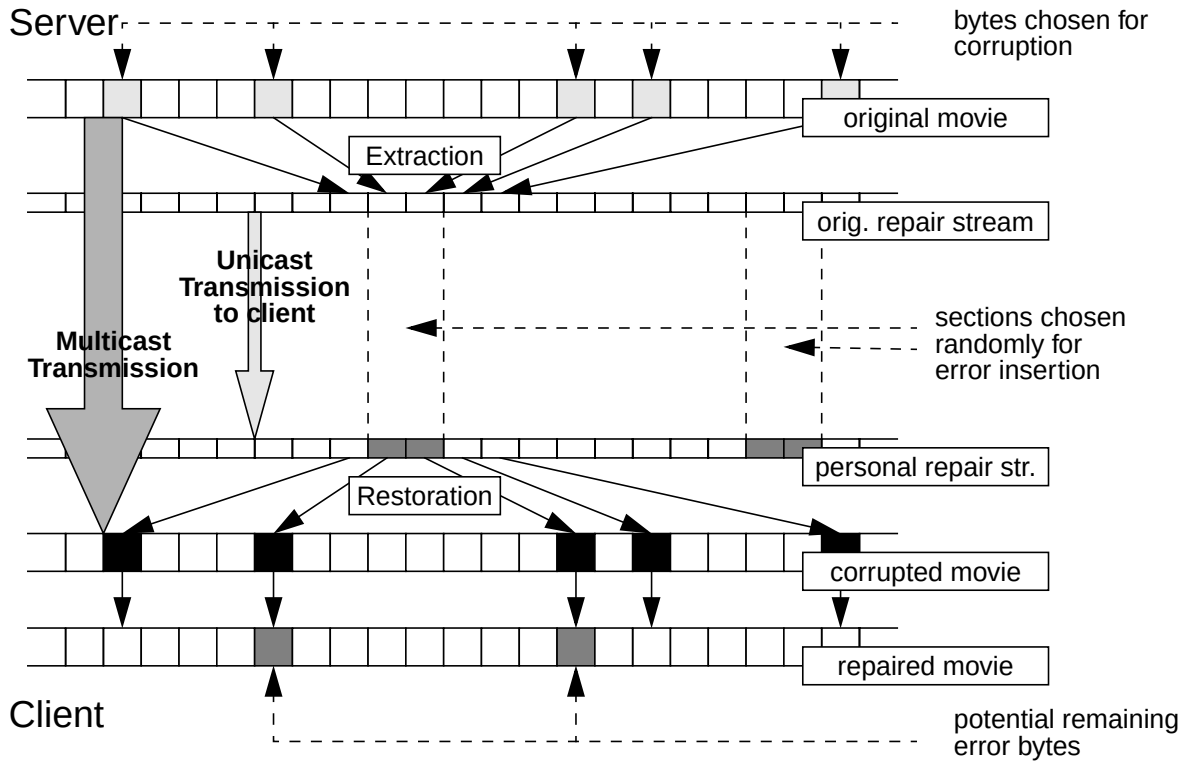


Figure 33: Random error insertion in Partial Corruption

intervals of the repair stream for each movie, into which errors are inserted randomly. Figure 33 demonstrates how this results in widely spread marked bytes in the repaired movie that is finally presented to the user. The movie-dependent, but customer-independent selection of section from the repair stream that may contain errors is done for the following reason: with the same total number of errors, the probability that collusion attacks fail can be raised when the number of potential positions of those error in the stream is reduced. At the same time, the errors are spread out widely and thus, the movie quality is not reduced in comparison to completely random placement of the errors.

The specific approach for error insertion works as follows: for each delivery of the stream, a uniform distribution is applied to put one byte error into each interval. Similar to the distortions of a watermark, each individual copy can be identified reliably by these randomly inserted errors when the provider keeps the random seed values in a database. For each copy of the movie, the bit error sequences can be compared with the series of bit errors which are generated by the seed value on file using a brute force approach.

If the attacker chooses a 3-party collusion attack to eliminate the bit errors, errors remain with some probability that can be used to identify at least two of the original customer. Let the length of the movie be S_f, the length of unicast portion S_u, with $T = S_f / S_u$. If the average offset is O and the length of each interval is I, there is a probability of $S_f / (OTI^2)$ that at least one byte error remains. For a 1GB MPEG-1 movie, 0.5% encrypted transmission, $O = 1000$ bytes (resulting in a $0.5 \cdot 10^{-6}$ byte error rate in the movie) and $I = 100$, this computes to a one byte error probability of 0.537. Smaller intervals increase this probability considerably.

Problems

The size of the repair stream is 0.5% of original movie's length, which is still a lot; to operate, the required point-to-point throughput between the content provider and each customer is about 1 kByte/s.

The partial corruption approach is based on the observation of the concept of partial encryption. It is mainly intended to reduce the effort of unicasting complete, encrypted streams (and decrypting these streams at the client side) while protecting from the theft of data during the transfer phase or from the service provider's cache, respectively. I.e., it is aimed at caching architectures. In the trivial setup, the customer would receive a broken stream, an encrypted patch stream, a decryption key, and finally after decryption and merging, an unmarked, perfect copy of the movie. The bit-destruction ideas were added in a straightforward way to fix this obvious flaw.

The bit-destruction approach is easily overcome by a single malicious service provider who orders decryption keys -or who observes his customers' traffic- until all bit errors have been removed by collusion.

5.8 Remark Approach to Marking and Secure Transmission

With the experiences from the previous techniques, we have developed a new technique called Remark - for its ability to insert marks for a customer and for a service provider independently. Remark is the integrated solution for commercial video on demand scenarios:

- It supports distribution hierarchies that make intense use of caching
- Its marking schemes identify malicious service providers and malicious customers independently; thus it does not require the content provider to trust its service providers or customers
- It enforces a contact between customer and content provider which permits logging and charging, but this contact is only cursory
- Its marking scheme, although based on randomness, is extended to prevent errors in significant bytes of the data stream

Remark is derived from a combination of the Partial Corruption and Chameleon approaches to handle both secure transmission with cache server support and copyright protection. It fixes the problems concerning throughput requirements that were noticed with Partial Corruption and the video-specific limitations that were noticed with Chameleon. Remark's copyright violator tracing scheme includes an identification of the cache owners (service providers) as well as the individual customers.

Operation

In contrast to the previous schemes, the content preparation of Remark is rather complex. Figure 34 is a sketch of the sequence of steps that are necessary. Starting with the original video, keys A and B that are unique to this video are selected. These keys are used to feed a random number generator as required by the Chameleon cipher. Another key C is chosen uniquely for the video, which is used for the corruption step according to the Partial Corruption

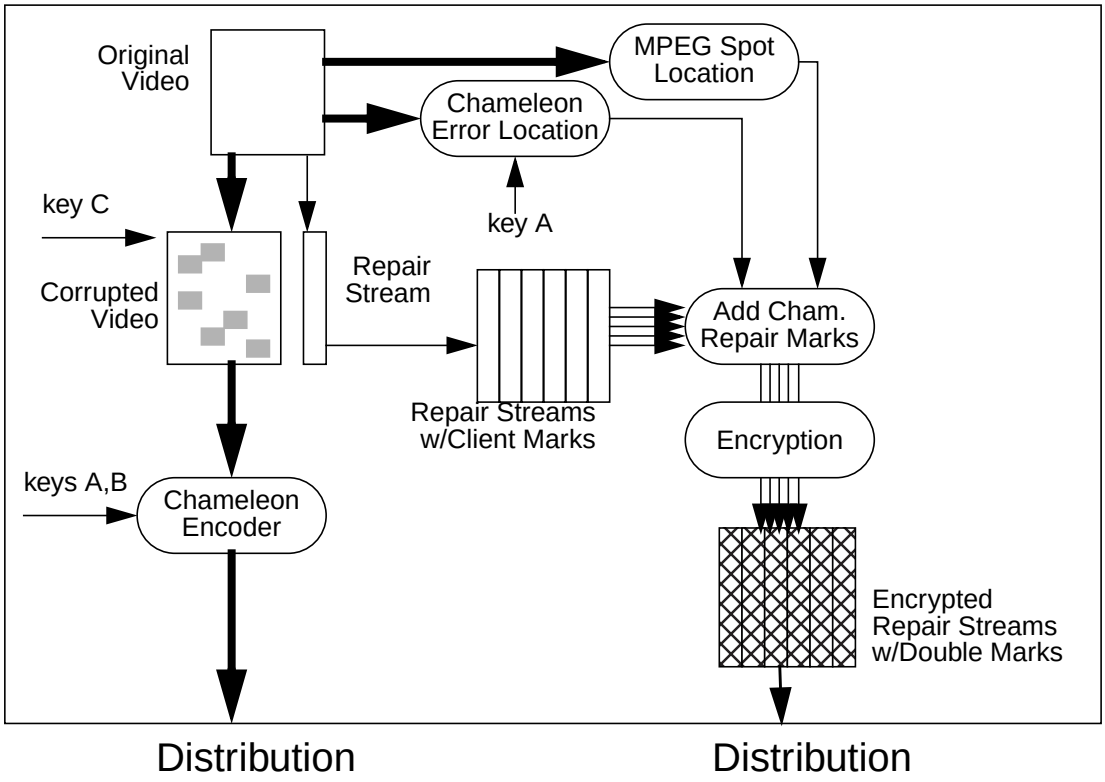


Figure 34: Content Preparation in Remark

scheme. Using key C, a corrupted movie is derived from the original movie, which is subsequently encrypted partially using keys A and B as in the modified use of Chameleon approach for videos. The resulting corrupted, encrypted movie is distributed freely.

Additionally, key C is used to extract the repair stream from the original movie, which is replicated an arbitrary number of times, and a fingerprinted scheme is used to introduce byte errors into each of these replicas (the result is labelled “Repair Streams with Client Marks” in Figure 34). The A key is used to identify the positions in the original movie that are potentially marked by any of the possible variations of the B key, an MPEG parser is used to identify positions in the movie that must not be modified to maintain an acceptable video quality. Combining these two position lists, a list of positions is extracted which must not be destroyed by the Chameleon decoder. For each of these positions, repair indicators are introduced into the repair streams; these repair indicators and the bytes of the repair streams are ordered to maintain the same order as in the original movie.

Each of these extended repair streams is than encrypted individually (with a key D randomly chosen for each copy) using a typical video encryption approach and are distributed to cache servers.

Client retrieval of the content is explained with Figure 35. A client who decides to retrieve a movie title contacts its closest cache server. The request indicates the movie, provides a public key to encrypt any responses, and a signature for potential proof of authenticity. The cache server adds to this request its choice of repair stream, its own public key, and its own signature; this request is forwarded to the content provider.[2] The content provider will store the request for potential proof of the transaction, than answers with the set of keys that is required for dis-

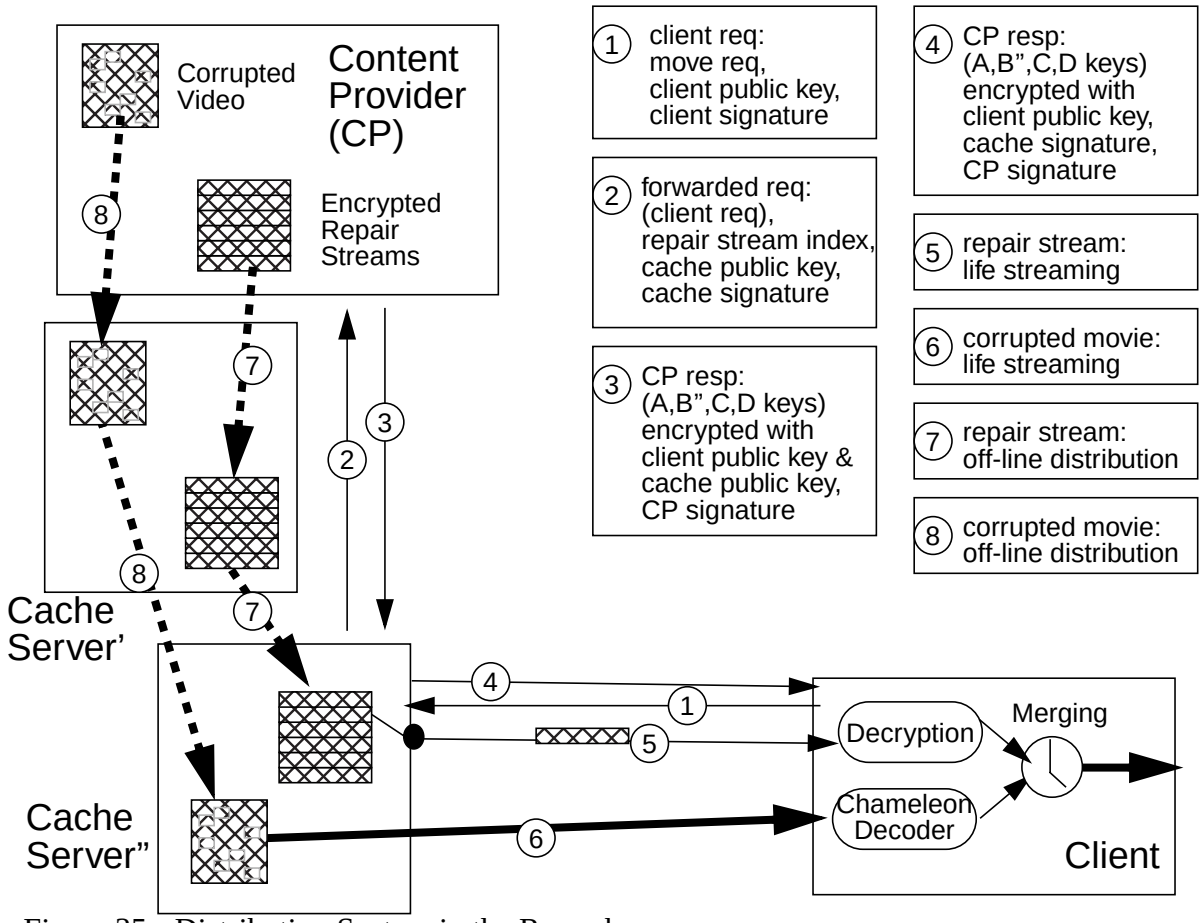


Figure 35: Distribution System in the Remark

playing the movie title: the D key for the chosen repair stream, the C key to identify repair stream positions in the corrupted movie, and the A and a cache-specific B' key to decrypt the corrupted movie. This set of keys is encrypted with both public keys and signed.

The receiving cache server applies the first decryption step to the set of keys and adds its signature. This is forwarded to the client. The client decrypts the set of keys and starts to retrieve, in real-time, the corrupted video and the selected repair stream from its cache server. When a repair stream has been transmitted once, it must be considered expended by the cache server, since the content provider will not re-issue the decryption key. Due to this restriction, a cache must monitor the number of hits on each movie title and request additional repair streams in due time.

The decisions that are taken in the decryption and merging process are shown in the pseudo-code of Figure 36. Key D is used to decrypt the repair stream. This stream contains two kinds of data: bytes that are required to replace the corrupted bytes at the position indicated by key C, and repair indicators for badly destructive modifications that may be introduced by the Chameleon decoding of the corrupted movie. The resulting video stream containing some non-fatal errors is forwarded to the presentation device.

2. This could be applied recursively to include all service providers that seperate the first cache from the content provider.

Protection features

Based on the results of Section 5.3, it is appropriate to extract randomly 1% of video data from a movie and to replace that data with random bytes according to the entropy calculation. Although the Partial Corruption approach allows other sizes than bytes, it is important that the destroyed units are very small (single bits or bytes) because the repair streams should be as small as possible with the decompression algorithm doing their work for them. Usually, the corruption of 1% of a movie's bytes should effectively confuse any MPEG-1 player if they are randomly chosen.

The number of customer-specific errors that are introduced into the individual copies of the repair streams can be tuned according to the environment. If the potential for collusion attacks is considered high, but the risk for theft low, it may be appropriate for an MPEG movie, to select bytes for potential destruction from P and B frames or from I frames' crominance data selectively, and to increase the error ratio in the displayed movie. If the opposite is the case, it may be more interesting to reduce the error rate in the displayed movie but make those errors more grave: identify primarily I frame data and P frame motion vectors and select errors that are located in this data (note that we assume that header can always be reconstructed easily).

```
i = generate from RNG1
j = generate from RNG2

do
   play up to min(i,j)
   if i < j
      get word W from SB
      compute next RNG1 XOR B XOR W
      compute feedback on B
      if (SF has ri)
         compute next ri
         get word V from SF
         play V discard W
      else
         play W
      endif
      i = generate from RNG1
   else if i==j
      discard next word from SB
      get word W from SF
      if (SF has ri)
         compute next ri
         play W
         generate from RNG1
         compute feedback on B
      else
         compute next RNG1 XOR B XOR W
         compute feedback on B
         play W
      endif
      i = generate from RNG1
      j = generate from RNG2
   else
      discard next word from SB
      get word W from SF
      play W
      j = generate from RNG2
   endif
while still data in SB
```

- `SF` - repair stream, with pre-generated errors, somehow encrypted
- `SB` - Bulk stream, broken and Chameleon-encrypted
- `i` - offset to next word that is affected by Chameleon
- `j` - offset to next byte that is affected by partial corruption
- `ri` - repair indicator in repair stream

Figure 36: Decryption in the combined approach

Without such additional considerations, the efficiency of the insertion of customer-specific marks is identical to the results of Section 5.7.

The Chameleon cipher for identification of the contributing cache server does not manipulate uncompressed content data in this approach. Rather, it manipulates the compressed data stream (except for the repair data). Thus, the bit error rate must be below 10^{-3} (which yields for MPEG-1, subjectively, an unaccaptable quality) and above 10^{-7} (which is considered 'repairable'). For a 1GB movie, this gives us between 10 Mio and 1000 potential bits.

The figures from Section 5.6 that evaluate the protection of the B key in the Chameleon approach indicate that it is not appropriate to reduce the key length or the number of marks from the proposed 2^{22} and 2^{12}, respectively. Since the algorithm generates an average of 2^{12} marked bits for 512 KB of potentially marked data with these numbers, a random choice of only 1% of all words of a 1GB movie (or 10 MB, respectively) for consideration in the marking process allows an average of 80000 marks to be introduced into the decrypted stream. Considering that this data is at least as well-protected from a collusion attack as the B key, this is an appropriate number of marks.

Since a less disruptive approach for the insertion of marks would be advantageous, some consideration should be given to this. For the time being, the removal of the worst effects of random marks in movies are addressed by Remark by the repair indicators. The common problem of Partial Corruption, Chameleon and Remark is the possibility to detect errors in the Huffman decompression step. The following may be a potential fix to this: by allowing the MPEG encoder to create only Huffman tables that are slightly weighted (for a limited penalty in compression efficiency). The Huffman tables are built in such a way that they have many sequences of identical length for different compressed representations which differ in exactly one bit. With a rather high effort for creation, this allows a repair stream that inserts personal "incorrect correction bytes" that are neither detected in the Huffman decompressor stage nor in any later stage of an MPEG player. A fingerprint is than represented by the sequence of decisions whether to use such a replacement value or not.

It may be necessary to add additional data for consistency checking to the repair streams. Otherwise, the merging function could be used to insert tracable junk data into a movie. The merged version of broken stream MS and junk patch stream would give away all potential locations of defective bytes, which decreases the effort of collusion attacks.

Evaluation

A variety of approaches has been considered and had to be discarded, since they were not applicable for the protection of a decentralized video distribution architecture. Our Remark approach provides a comprehensive means of affordable but secure transmission and of copyright violator tracing in a multilevel distribution system. Its marking schemes identify malicious service providers and malicious customers independently. The basic design does not include marking of more than one service provider in one delivery. However, the system can be operated in a hierarchical manner as well by applying additional encryption and signing steps.

Remark is scalable with respect to the number of service providers and to the number of users served by each provider. It is very resource-saving for a delivery system that includes

end-to-end communication - which is does by design to permit logging and charging at the content provider's site. Obviously, the client needs a broadband connection to its cache server, but without such a connection, there is no need to consider anything like video-on-demand. The wide-area throughput requirements are much lower than this. The combined size of all keys that need to be delivered from the content provider to the client on-demand is well below 1 MB. Considering a five-minute unencrypted preview phase for a movie, this gives an average end-to-end bandwidth requirement of 3.4 KB/s which is available in the most parts of the Internet today.

Neither does it impose any special requirements on the design of the transport protocol, nor on the server design. Synchronization that is necessary for the merging of corrupted video and repair stream is easily solved at the client side using established synchronization approaches such as [JG97].

6. Simulation Modeling Basis

The number of aspects that can be potentially considered for a wide-area distribution system is large. Due to considerations presented earlier in this thesis, we take only distribution options into account that include caching and de-centralized management. Alternatives would be investigations in other areas such as multicasting ([KVL97], [GLM96]), client-caching and forwarding ([HCS98]), scalable best-effort streams, etc. This study deals with video caching because it is a path that has been taken with low quality video on the Web and that may grow into a commercial business when higher qualities can be supported.

Since the overall goal of this thesis is the design of a wide-area distribution architecture in general, the requirements for an investigation have to be refined. We can assume that all kinds of architecture are able to perform in some way. For example, we are aware of the knowledge advantage that a central architecture has implicitly in placement decisions when centrally controlled caches are used. We will not try to investigate the error resistance of our architecture - such investigations are currently undertaken e.g. in [SAW97]. We assume that we can neither control the topology of the network nor the placement of cache servers. We try to identify a well-balanced pair of distribution mechanism and replacement strategy for a decentralized system. It must be applicable to a wide range of predefined network bandwidth/server size combinations to remain valid for some generations of network and server technology. Thus, our metric must allow a qualitative evaluation of the following requirements:

- small number of service declinations by users due to waiting time
- small number of service refusals due to network overload
- resistance to variations in movie popularity due to the time of day
- resistance to local variations of user interests

Besides these criteria -which can be represented by a single number, the percentage of successfully requested movies- we need additional values:

- percentage of storage and network usage
  This indicates whether and in which way an example system in our test was over- or under-dimensioned.
- average time between caching and removal of a title
  This indicates stability of a caching strategy's decisions.
- cache hit ratio
  This indicates quality of a caching strategy's decisions.

Since these are indicators that a strategy can not be considered fully performance-tested, these values are an important part of the performance testing but they are not part of the performance metric. We are using them as indicators of the direction that our investigations have to follow subsequently.

These constraints as well as abstractions that we have introduced into the model of the physical distribution system are presented in Section 6.1. The traffic that is generated in such a distribution system by user requests on inserted content is considered in 3 sub-sections. Section 6.2 presents and evaluates existing work on workload modeling that could be found in the literature. A specific issue related with existing workload models is the application of knowledge

that is derived from the user requests' property of being Zipf-distributed. The investigation of the applicability of this distribution is presented in Section 6.3. This is followed by a section that argues for separate models for movie life cycles and for user requests on such a model, and that presents our models. Section 6.5 introduces the problem of variations in movie popularity throughout a single day, and our approach as evaluating their effects in the absence of real-world data. The simulation program that builds on this knowledge is presented in Section 6.6. Finally, the state of this model is evaluated in Section 6.7.

6.1 Topological Considerations

This sections presents a collection of considerations that have influence on the topological modeling in a wide-area VoD simulation. Several characteristics of VoD traffic need to be accounted for, respectively can be exploited, in our simulation. We consider the following constraints valid for VoD systems in general.

6.1.1 VoD Characteristics

The following characteristics are specific for video-on-demand, i.e. they apply to all kinds of video distribution, independently of the distribution architecture.

Access Patterns

In a VoD system, we expect movies to be typically consumed once per request, and from start to end without major requirements to seek or rewind operations.[1] We restrict our investigations to non-interactive access patterns.

Read-Only User Access

Although VCR operations may occur, all requests by users are read-only requests. We expect that this allows much simpler strategies than in other cache-exploiting system, e.g. in distributed databases, to yield adequate results.

Read-Write Owner Access

In our model, we can assume that only the origin of the information can change or update it. Since we expect that this is very rarely performed, we exclude it from our model. This is in contrast to news-on-demand systems, which should be able to revoke news after publication, or to update figures that are included.

Delayed Replacement

Furthermore, the situation is less difficult than other environments where synchronization of multiple copies is an issue. Although it may be questionable from the legal side that copies in caches are replaced or updated at some time after the master copy, this does not seem like a vitally complex issue that needs to be reflected by the simulation - mainly because such updates are extremely rare for movies.

1. Observations in VoD field trials -that are to my knowledge undocumented- indicate that seeking is a rare operation, but that the movie start is a typical candidate for fast-forward operations.

Large Objects

We assume that objects are large and that loading times are large as well. It may be appropriate in a real system to perform a bulk transfer from a higher level cache server or the original server to the lower level cache servers, rather than using streamed transmission. However, in a large scale decentralized infrastructure, load-balancing web servers that use throughput measurements, bandwidth sharing and other streaming mechanisms rather than plain TCP-based file copy have proven their effectiveness; we assume therefore that controlled streaming is an appropriate approach in a decentralized caching architecture; especially in conjunction with techniques such as DiffServ ([RFC2475]) that allow "better best effort" transmission from upstream server to cache.

Our model assumes thus streaming, which allows distribution mechanisms to exploit the additional advantages of multicast transmission for complete or partial movies.

6.1.2 Characteristics of VoD with Caching

In the following, several real-world implications of the use of VoD in a cache-based distribution system are discussed.

Limited Cache Size

Our caches are supposed to be small w.r.t. to the size of the objects that are cached. Although the increase of available bandwidth and the decreasing price for storage space implies for some people that limited cache size will not be a problem any more in the near future, we believe that this is not the case. Rather than this, the resources will be consumed by a increase data size that is demanded for on increased quality for several hardware generations. Today, a 72 GB disk[2] can hold approximately 1,000,000 web pages, which is sufficient for only 100 average MPEG-1 movie or 12 MPEG-2 movies. This implies to us that complete replication of all titles is not an issue right now.

Complete Caching of Objects

Although our focus is on streamed media, we do not consider the case of partially keeping a video in a cache server. This would be done in case of an approach that is intended to decrease the startup latency, while the bulk of the video data is delivered from a remote server. Our setting assumes that servers operate on complete videos because we assume unpredictable jitter, packet loss and server downtime.

Write-through Caching

In contrast to web caching techniques, we do not apply store-and-forward of complete videos, which would add ridiculous delay before delivering the data to the end user. A solution would be to use store-and-forward transmission of blocks of the video. This would add only a limited per-hop delay to the transmission and could be exploited.

This contradicts our decision to store only complete videos. The assumption that video caches will typically not be installed on routers but rather be installed on dedicated machines

2. http://www.storage.ibm.com/hardsoft/diskdrdl/ultra/72zxdata.htm

inspired a different solution. Assuming that the cache needs to communicate with both original server and client through a router or firewall (and potentially using the same network interface in both cases), this doubles the amount of data that a server (and the network) has to handle in forwarding compared to receiving the data only. Further, packet loss and jitter can be hidden by the cache only if some undesirable delay is introduced before the data is forwarded to the client.

Since the currently available video servers tend to use RTP for the transmission, we consider it sensible to accept the additional jitter and packet loss, reduce the delay, and transmit the video using RTP to transfer to a multicast address that both client and cache join. A compatible extension to RTP for reliable transfer of the object to the cache has been implemented to make this approach feasible and is documented below. Using the multicast approach, the cache server's interfaces (and potentially, the network to which it is connected) experience less load per cache miss and the client experiences less delay in case of a cache miss.

Cache Servers Located off Routers

We assume that cache servers are typically not located on routers for various reasons - although this contradicts probably the philosophy of router manufacturers. Routers should be optimized for passing packets from one network interface to another with as limited a delay as possible. Video servers, on the other hand, reserve resources for scheduling the CPU, the memory or the disk in order to optimize data retrieval and playout.

In spite of this assumption, we simplify the model by identifying cache servers with routers. This is considered a permissible simplification because we have presented LC-RTP in Section 4.5, which allows write-through caching in such a way that a cache that is located off the router will listen to the video stream at the same time as the initially requesting user receives the video stream. We have also proposed a use of the RTSP protocol that allows piece-wise delivery in such an environment, assuming that the client is capable of some synchronization.

This does not violate the assumptions that we made for write-through caching since our server throughput model covers a situation where the cache server is located in a network which is attached to the router and through which the data stream is transferred once in each direction. Figure 37 shows the simplifications that are made in the model.

6.1.3 Structural Simplifications

The topology that we use for our simulation model is a simplified view of a distribution system. Rather than assuming that an original server is located in an end system or that the original network is not a backbone, our model assumes a hierarchy that places the original server into a top-level backbone network.

On this basis we consider ways to deliver individual video streams from a content provider through a broadband network to a customer. We do not distinguish whether the network is shared with other traffic (as might be the case with the Internet as a backbone of this system) or whether it is a network that is dedicated to the video-on-demand system (as might be the case if current cable television infrastructure is recycled to these ends), because we assume that in both cases the required resource capacities in the network are sufficient for streaming the video in real-time if a transmission is actually started.

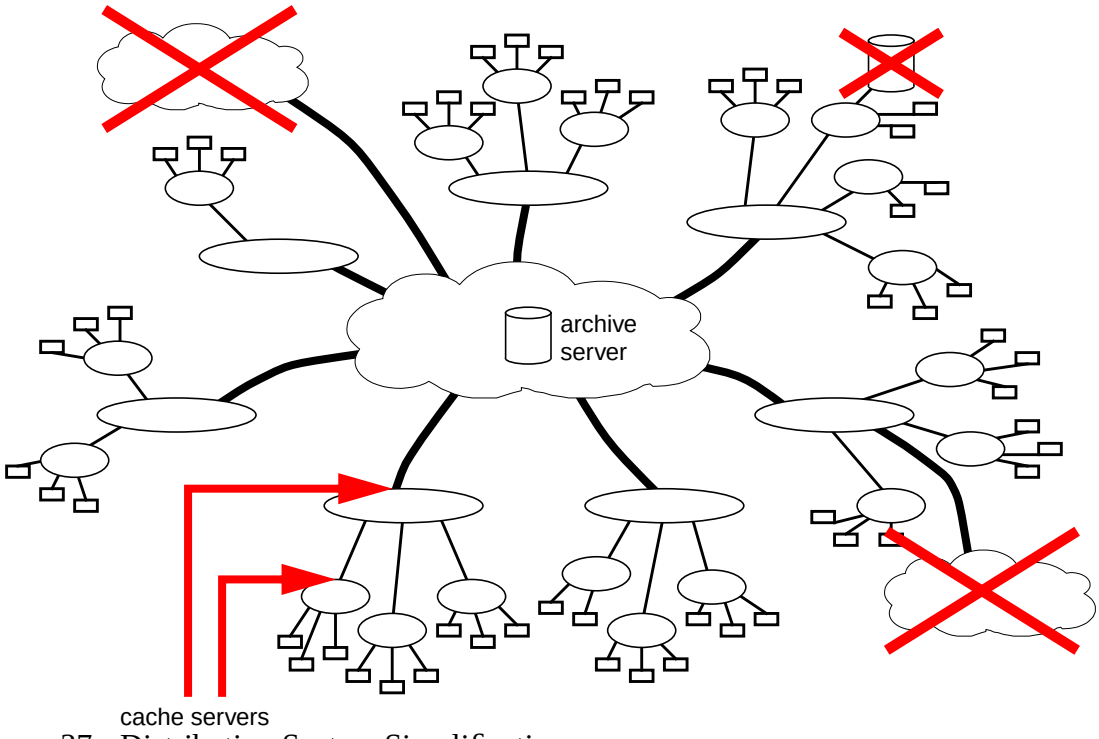


Figure 37: Distribution System Simplification

Network Overload Detection

This condition is not entirely compatible with the other assumptions that we make in investigating an Internet-style network for video distribution. We do not make the assumption that cache servers are gateways or routers at the same time, either. This makes it technically difficult to assume admission control for the distribution system.

For a simplification of the simulation, however, this assumption is very valuable, as well as for the detection of network overload. Furthermore, it is conceivable to implement this mechanism as an enhancement to a distribution system, e.g. one that is based on RTP using IP-multicast: since cache servers communicate hierarchically and the clients communicate only with their assigned cache server, those caches can collect information about the network status and refuse to deliver streams due to network overload as soon as they receive notifications of major packet losses at their clients, e.g. in case of delivering with RTP by listening to RCTP receiver reports.

No QoS model

We would expect a VoD system to exploit resource reservation option in the network as soon as they become available. Specifically with our protocol, LC-RTP, we assume a relevant performance increase with reservation mechanisms of the IntServ ([RFC1633]) or DiffServ ([RFC2475]) approaches.

In our simulation, we simplify the model by assuming admission-controlled transfer of streams, since a streamed transmission needs to rely on the availability of bandwidth. We do this without explicitly modeling the time that would be required for bandwidth negotiation. It

is assumed that the overall delay can be used as a model for such signalling activity that is necessary before the actual start of transmission.

Ignore Indirect Routes to Caches

We can expect that networks exist where users have redundant connections to several cache servers, or where cache servers themselves are not organized in a tree. For our delivery structure, we ignore this possibility. The reason for this simplification is the assumption that a preferred delivery path arranged in a tree structure will be used for delivery under typical network conditions, i.e. without overload or connectivity loss.

Local Delivery Decisions

There is no technical need for immediate forwarding of the access information to the origin of the information. Such a need is introduced by security concerns and the requirements for copyright protection. However, in Chapter 5 we have presented a means of reducing this communication with the origin of the movie to an amount that is hardly relevant in comparison to the data transfer rate. In the simulation, we will neglect the forwarding of information to the origin unless demanded by the investigates strategy.

6.2 Overview of Workload Models

Various approaches towards modeling the load of video servers have been proposed in the literature. Usually, they are not designed for simulation models which consider the amount of interaction that takes place from the user's point. Rather, an analysis is performed to derive the worst case situation that a server (or network) can cope with.

6.2.1 Reasons for Workload Modeling

Workload characterization involves studying the user, network and server environment, their key characteristics, and the development of model that can be used with a variety of parameters. Once a workload model is available, the effect of changes in the workload and system can be evaluated by changing the parameters of the model. Workload modeling is required in a performance analysis, if the investigation is analytical or if it uses simulation.

If the system exists already, an alternative approaches for an investigation are measurement and the use of trace files that capture real workload as an input for a simulation. In case of VoD, the field trials that have been implemented in various countries world-wide would perhaps allow the use of traces. However, no such traces are made available. Alternatively, traces from applications that are considered similar to VoD traffic can be applied. The following sections present techniques that have been applied and the restrictions of these models.

6.2.2 Modeling for Momentary Load

One approach is the modeling of single video streams as they are accessed and played. This is generally done in order to understand how the operation of a single machine or cluster of machines can be optimized. Little and Venkatesh take this approach in [LiVe94b] with the goal of optimizing disk I/O operations in a single system. Their approach is to build an analytical

model for access probabilities based on the work by Ramarao and Ramamoorthy [RaRa91]. In [TMD95], Tewari et al. optimize the I/O utilization in a server cluster and use Poisson processes to model the user accesses to the server, with the mean value chosen according to Little's Law [Litt61]. Golubchik et al. investigate in [GLM96] means for sharing video streams in a video-on-demand systems that holds when VCR controls are permitted to the user. Their user model is analytic and assumes a Poisson arrival process. Their goals do not require any understanding of long-term movie development.

These approaches are useful for optimizing playout, stacks or disk operations in a system, but neither take user interaction into account once that a movie is playing, nor do they try to model the play time of a single clip in any way.

The modeling of VCR commands requires a model that includes modeling of user habits in applying these VCR operations. In [DaSi95], Dan and Sitaram analyze the caching of data in a single servers or server cluster under various interactive workloads and models information such as access skew (the distribution of requests on stored clips), the clip length distribution and the viewing time. While they do not consider the aging of individual movies, since their need is for a short-term model, they consider the distribution of hits on the available videos and chose the Zipf distribution to model the video popularity. This Zipf distribution is examined in Section 6.3. [Cher94] has proven that this formula applies to popularity distributions among videos. It is noteworthy that this distribution, which is typically used as the basis for investigations on video server operations, is completely independent of the number of users that access the set of movies.

Nussbaumer, aiming at optimizing caching in a single server system or cluster, also assumes the Zipf distribution [Chen92] to model video popularity [NPSS95]. The distribution of videos or blocks of videos over multiple machines for load-levelling or availability purposes has been investigated in [BeBi96].

Barnett et al. [BAB95] aim at minimizing the storage costs in a distributed system and apply caching mechanisms to do this. This requires the kind of long-term analysis we also discuss in this paper. They base their considerations for long-term popularity (in the absence of freely available video-on-demand trial results) on numbers from CD sales. The model they derive is a double exponential curve for the distribution of user accesses on videos and a movie popularity development with only one raising and one declining side. They evaluate various caching strategies.

6.2.3 Modeling Variations from Day to Day

In previous models, the issue of day-to-day variations in video title popularity was not addressed in any way that can be sufficient for our scenario. The reason for this is that previous work dealt mostly with short-term problems on the order of seconds at best that have been addressed by other studies in the video-on-demand field.

It has already been shown by Barnett [BAB95] that the Zipf distribution, being static in time, in itself is not well suited to simulate long-term developments. Because of this, it is not applicable to investigations that consider temporal changes.

In order to compensate for this and to add to the Zipf distribution a long-term dynamic change in time, Dan and Sitaram [DaSi93] have created a model based on a modified rotation

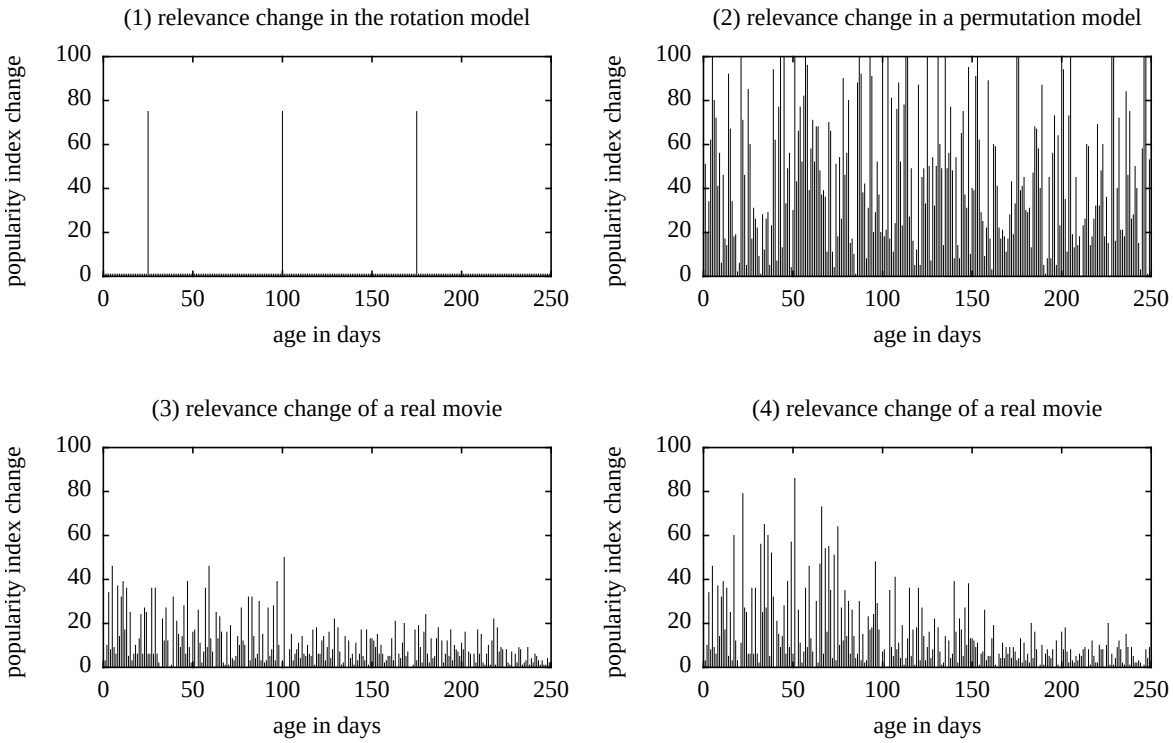


Figure 38: Day-to-day relevance change measurements

of movie rental probabilities. They take into account that the distribution of movie rentals at any time can be approximated by the Zipf distribution but that the ranking among the movies is changing in time. They assign an index number for the Zipf distribution to each individual movie title, and its current popularity is calculated by the Zipf distribution, the movie with the lowest index number is the most popular movie of the day. After a fixed amount of time, new values are calculated by rotating the indices. To reduce large jumps in the relevance of a specific movie, the left half of the movie indices is swapped before and after the rotation. By adding the rotation, they try to simulate dynamic changes in the rental probability of individual titles for the cache of a single server.

They did not create this model for long-term variations and it is not applicable in that case. The graphs in Figure 38 demonstrate the drawbacks of this shifting approach that become visible when the amount of change in relevance is observed. The figure shows also another variation of using the Zipf function that simply permutates the indices. Each of the graphs shows the absolute change in relevance that a movie experiences from one day to the next. Graph 1 in Figure 38 shows this for the rotation model, assuming 150 movies. The comparison demonstrates that this model does not provide sufficient realism for long-term considerations when the movement of movies between cache levels is an issue. Graph 2 shows the relevance changes for a system with 150 movies that assumes a daily permutation. In comparison to these two models, graphs 3 and 4 provide two examples of relevance changes for real movies from a movie rental store with a small user population.

The comparison demonstrates that an algorithm which calculates the location of movies' copies in a distribution tree with respect to relevance can not be verified with either of the two models.

6.2.4 Conclusion

We have examined the workload models that were used in several video-on-demand papers and found that they have been typically developed for investigations concerning server performance, and that they have been used for investigations on the scale of several seconds to a few minutes. If a caching strategy does not account for day-to-day variations at all, videos will most probably be shuffled back and forth among small caches close to the end user when the strategy is implemented in the real world. The important issue is that day-to-day variations do neither reflect on large user populations nor on central server systems. They exist primarily due to randomness in user access patterns and not due to popularity changes in the movies life cycles. Thus, we have concluded that a new model needs to be developed.

6.3 Problems of Workload Models Based on the Zipf Distribution

A lot of earlier work is based on the observation that hit rates behave according to the Zipf distribution, and that caching, prefetching etc. can rely on this. This distribution is defined as

$$z(i) = \frac{C}{i^{\zeta}}, C = 1 \Big/ \left(\sum_{i=1}^{n} \frac{1}{i^{\zeta}} \right)$$

where $i \in \{1 \ldots n\}$. This is the original definition of the Zipf distribution, presented to describe the distribution of word lengths in the English language. In video server documents, a subclass of Zipf functions is often used referring to the PhD thesis of Chervenak, which assumes $\zeta = 1$, i.e. the typical assumption in the video on demand literature is

$$z(i) = \frac{C}{i}, C = 1 \Big/ \left(\sum_{i=1}^{n} \frac{1}{i} \right)$$

In the formula, *n* is the number of available movie titles. *i* is the index of a movie title in the list of *n* movies that are sorted in the order of decreasing popularity. It is one of validations for the 90:10 rule of thumb for popularity distributions.

To verify the applicability of the distribution, we compare it to our own data which originates not in a movie magazines but which is the anonymized data of 2 years of a local rental store with a small number of customers. We compare all days of one month in the period covered by the movie data and sort 250 representative movie titles by their popularity at these days[3]. The resulting data is compared with the Zipf distribution for *n=250*. Figure 39 shows the first 100 entries of the resulting curves. It presents the curves for the two days with the lowest and highest hit rate on the top 10 movies in one month in comparison with the Zipf distribution for the same number of movies. The month was chosen randomly.

This demonstrates that the Zipf distribution with $\zeta = 1$, although quite similar to the actual rental probabilities, is somewhat optimistic, at least for small user populations. The upper curve that was derived in this month may be restricted by the number of copies available in the rental store. The lower curve is not affected by this but shows that the diversity in user selection is wider than accounted for by the theoretical function.

3. Movies are counted per rental and day, i.e. one rental for 2 days counts 2, 2 rentals in 1 day count 2. Movies are typically rented for less than a day due to the price structure.

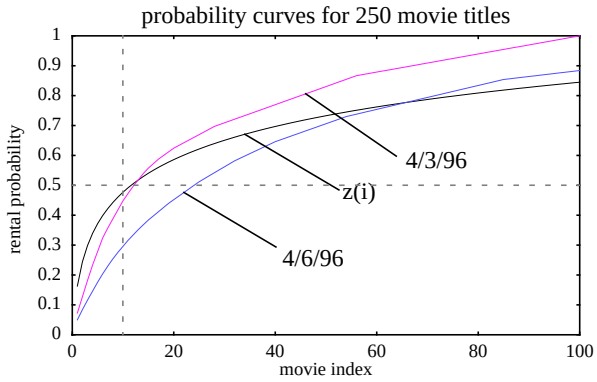


Figure 39: Rental probabilities compared with the Zipf distribution

Marshall and Roadknight have expressed their doubts about the use of this distribution for web traffic modeling in [MaRo98] on the basis of anonymized web cache traces. We have reservations concerning the application of this rule for system evaluation as well. While we will not disagree with the observation of Zipf behavior in VoD scenarios -our video rental data is not in open disagreement-, we have problems with the creation of a long-term workload model on this basis - something which was not needed for the investigations of [Cher94]. The following short analysis of hit probabilities is intended to demonstrate why we consider it unfit as the input of a model.

For simplicity, we assume a fixed distribution of movie access probabilities in accordance with the Zipf distribution - i.e. we make a snapshot observation of an access distribution. We consider this legal because the change in movie popularity is a long-term effect. Then, we compute the hit probabilities for each user in a population at that time, and compute the average deviation of the access probabilities from the mean.

Our observations support the position that a Zipf distribution is well suited to describe observed popularity distributions. However, we do not believe that this distribution can be used to generate user requests. The first argument in our favor is given by the standard deviation of each movie *i*'s probability *z(i)*. Let us assume that *N* movies are distributed according to a Zipf distribution with $\zeta = 1$. For clarity, we call movies with indices *n* and *n+1* for an $n \geq 0$ according to the Zipf distribution *neighbors*, and movies with the lowest indices the *most popular movies* in the following explanation.

We define the random variable X_i^U as the portion of hits on the movie *i* when *U* independent draws are made from a set of *N* movies. The expected value of this random variable is $EX_i^U = z(i)$. The variance of this of X_i^U is $\sigma^2 = E(X_i^U)^2 - (EX_i^U)^2$, where

$$E(X_i^U)^2 = \sum_{j=0}^{U} \left[\left(\frac{j}{U} \right)^2 \cdot \binom{U}{j} z(i)^j (1 - z(i))^{U-j} \right]$$

When this is used to compute the coefficient of variation σ / EX, and compared to the relative change of hit probabilities between neighbor movies, we can observe that the deviation from the Zipf behavior is hardly relevant for the most popular movies. But even for moderately large user populations, this deviation exceeds the relative difference in hit probabilities between neighbors in the Zipf series. Figure 40 tries to demonstrate the problem by illustrating the coefficient of variation and the relative popularity difference between consecutive movies. The graph allows the following observation: the deviation from the expected value is more impor-

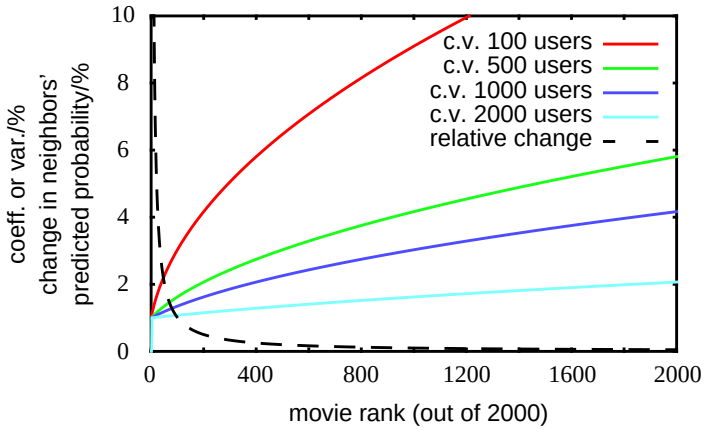


Figure 40: Illustration of Zipf neighbour change vs. coefficient of variation in hits.

tant than the order of the items that adhere to the Zipf function. This in turn implies that the Zipf distribution may be a good means for observing reality, but it is not a sufficient basis for modeling reality.

In [BGW97], we had assumed that this was a problem for small user populations; however the exponential property of the relative difference in the Zipf function indicate that this problem will require extremely large user population to apply successfully a 90/10 or even an 80/20 rule-of-thumb.

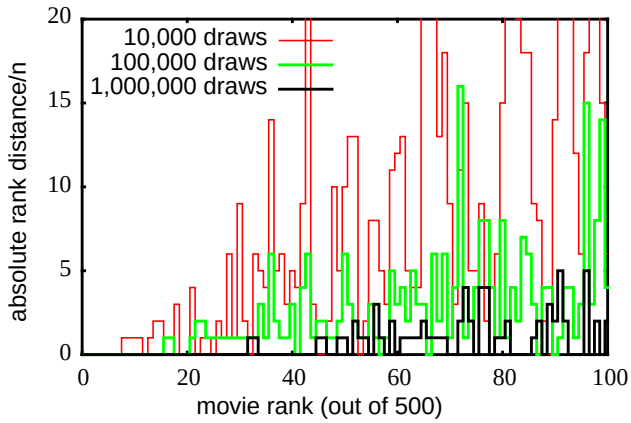


Figure 41: Deviation of experimental from predicted order of popularity

To clarify this problem even further, we generate probabilities for an ordered set Z of movies according to a Zipf function (see Figure 39). We randomly draw movies from Z according to the distribution; we repeat this experiment for 500 up to 1,000,000 draws. The time frame that we are covering with these draws is irrelevant - it must be limited to a sufficiently short interval that popularity changes of the movies due to aging (i.e. their life cycle development) can be ignored. We try to find out how well these draws fit with the predicted ordering. We create another ordered set $\bar{z}$ of the same movie, with an ordering according to the decreasing number of actual hits in our experiment.

For the first 100 movies of Z, Figure 41 shows the absolute distance between the movies' rank in Z and $\bar{z}$. Only the 100 most popular movies (with the lowest indices) are shown in this figure, and the largest distance shown has been limited as well. For the movies with higher indices, a system designer would not care because rarely requested titles are not expected to be

cached. The distance is increasing for those movies. The most popular movies are sorted as expected; for slightly less popular movies, the actual and predicted position is diverging quickly unless the number of draws is huge. The figure demonstrates that the deviation of the actual hit counts of a movie is relevant even for large numbers of draws. It demonstrates also that there is a relevant gain in knowledge about a large number of selection.

The assumption that the Zipf distribution is still a good means for describing hit distributions is supported by Figure 42; Figure 42 (a) is a graph of relative hit probabilities of Z and $\bar{z}$. They fit perfectly, which indicates that $\bar{z}$ is, in spite of a different order, still observed as a Zipf distribution. Figure 42 (b) underlines this by showing the differences between the relative weights of Z and $\bar{z}$.

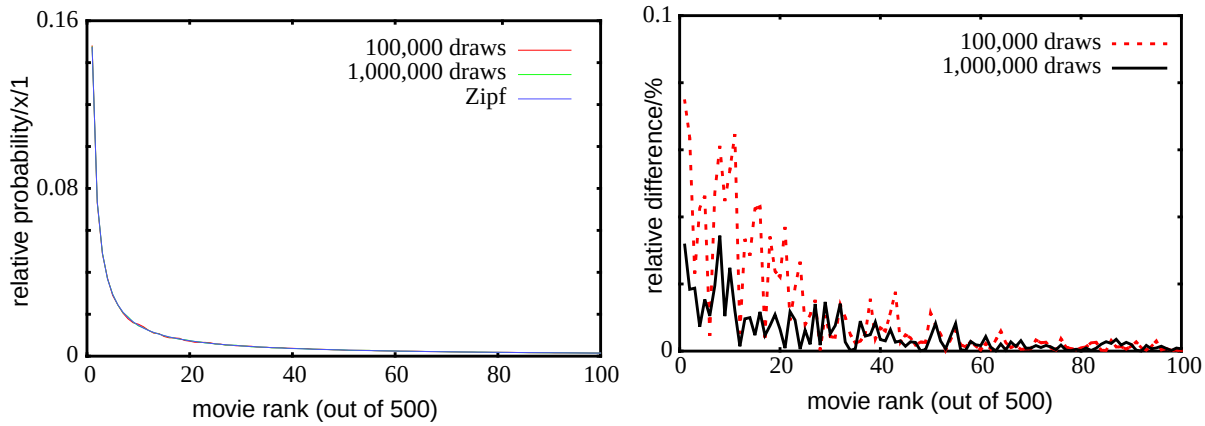


Figure 42: Comparison of predicted and observed ranks

Conclusion

We agree with other authors that the Zipf distribution can be used to describe observed user request distributions. Even when the user population is small, the identification of the top titles is probably good. However, even when the user population is large, the prediction for the average titles is bad - it may be perfectly appropriate to a replacement strategy as simple as LRU with as good results as a much more complex strategy. The number of titles that have a good chance of being recognized is growing, slowly, with the user population size. Under the condition that this observed property of the Zipf distribution of movie popularity is correct - which we do due to the data of [Cher94] and [BGW97] - this implies that gaining information about hit probabilities is advantageous for caching strategies.

For modeling of the user behavior at larger time scales that are relevant to our distribution system and replacement strategies, the movie popularity's property of being Zipf-distributed can not be exploited. It lacks the time-dependency, and it ignores the immense differences in hit rates between caches with a small number of users and the complete user population. In the following, we will separate these two issues to create a better model.

6.4 Separation of Movie Life Cycles and User Behavior

Our approach is to distinguish between movie life cycle modeling and user modeling. The behavior of users who want to see a movie and the development of individual movies seem to be decoupled in reality. This may reflect the situation that a user's decision to watch a movie is

based mostly on the available spare time rather than the existence of an interesting movie. For our model, this implies that the insertion of a relevant number of movies into our model at any one same time does not generate any increase in the overall number of rentals. Although all new movies are in the most popular phase of their life cycle the number of accesses to them remains low because the number of users, respectively the time they have available for viewing, is limited.

The independent modeling is also more convenient for modifications. It allows, for example, the experiments that concern daytime variations of movies' popularity. In the basic case, we assume that there is no reason for a user to arrange his own schedules according to movie timetables. Thus, the time at which a movie is retrieved is completely independent from the choice which is based on the movie's popularity at the time of retrieval. Since we have not had any relevant data on daytime variations of users' interests available, the daytime variations introduced in other simulations is modeled by a simple sinus curve on top of the movies popularity.

6.4.1 Demonstration of the Population Size Effects

We have two sources of data available for the video access information that we can use to validate our model. On the one hand, there is the VideoWoche magazine, which presents statistics that are derived from the weekly input of some hundred video rental stores, and on the other, the (anonymized) databases of a single video rental store. The available data covers roughly the same time, 1995-1997. The magazine's content limits the information that we receive to information on the movies that are considered the top 100 country-wide.

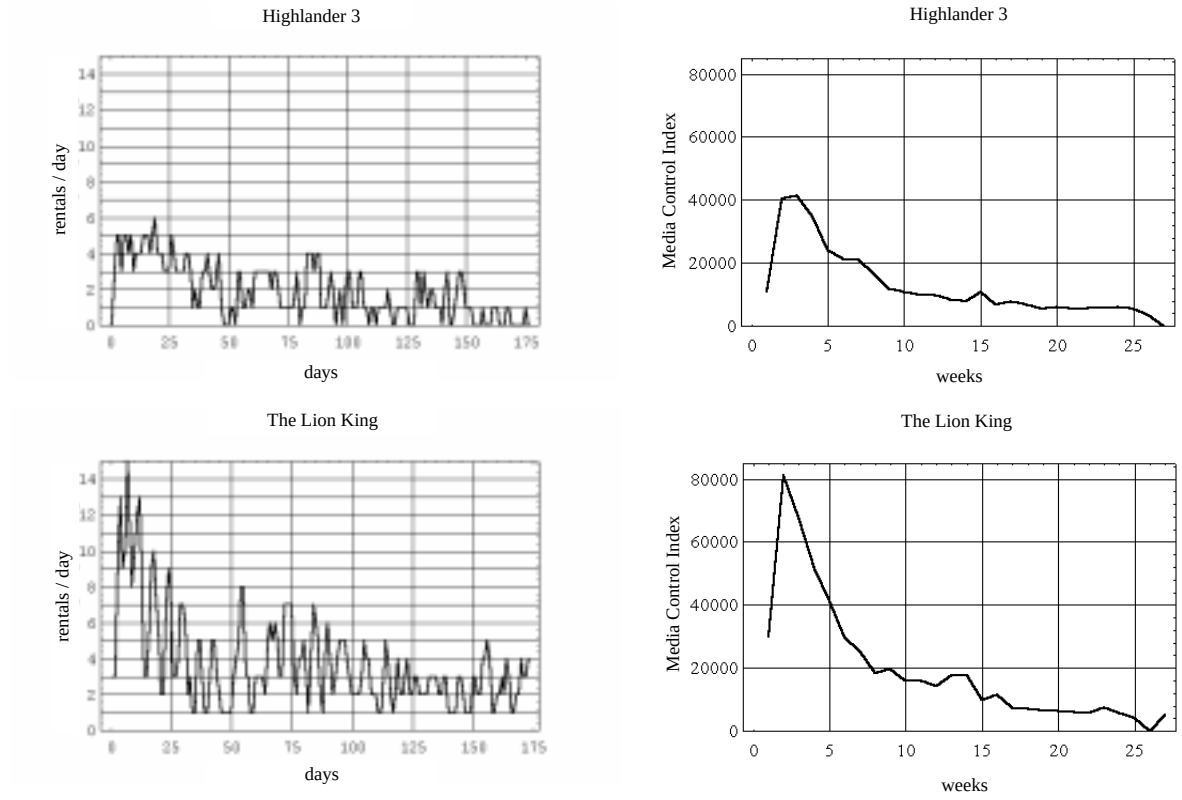


Figure 43: Comparison of rental store (left) and magazine numbers (right)

Figure 43 shows the comparison of the data from VideoWoche magazine (right) and the curves of the rental store (left) for two movies (Highlander 3, Lion King). We have chosen these two movies as examples because they were the ones that experienced the largest number of accesses to themselves of those movies that remained in the top 100 list of VideoWoche for several weeks during the observation period. If less popular movies had been chosen, the similarity in trends between this example data and the magazine could have been illustrated only by smoothing the rentals, e.g. by showing the average rentals in three days. By selecting these movies, the similarities become visible without any smoothing, while the day-to-day variations of the single shop remain easily visible as well.

We find that such figures supports our assumption that the number of users who access a provider (which we call its user population) has a major influence on the smoothness of a single video title's development. The size of the user population in a hierarchical system is not only relevant as an overall number. The observed increase in the variation of movie popularity for small user groups can be relevant for distribution algorithms.

6.4.2 Long-term Life-Cycle modeling of Movie Life Cycles

Newly published movies exhibit typically, but not always, a steeply rising start peak of user interest. The observation of the rental behavior shows that all movies share a general decrease of user interest in them, but this decrease is not identical for all movies and it is frequently interrupted by increases of user interest. Once a movie has been inserted into the system, its rental probability will never return to zero. On the contrary, the relevance of old movies can be quite high. We did not make a detailed category study, but marginal checks showed that, e.g., the start peak is less relevant (although existent) for movies rated PG-18 but the sustained relevance remains generally high.

In our approaches to create a model that reflects the long-term behavior of real movies, we had experimented with discrete models to integrate the rental behavior as reflected at a rental store. These models were discarded because they needed too many parameters with no explanation for their necessity.

Splitting the available data into a an underlying curve for long-term behavior of a movie and a random effect that is mainly dependent on the size of the user population led to a more appropriate model. We observed that the underlying curve seems similar to a variation of the exponential curve that is used to describe, e.g., the spread of infectious diseases. We used a parametrized version to take into account the quantitative difference in the number of rentals, the steepness of the loss of interest and the remaining interest in a title. This function is

$$RP(t) = a \times e^{\left(\frac{2}{\sqrt{10}} - \frac{t}{10 \times b} - \frac{b}{t}\right)} + c$$

Figure 44 shows what this function looks like.

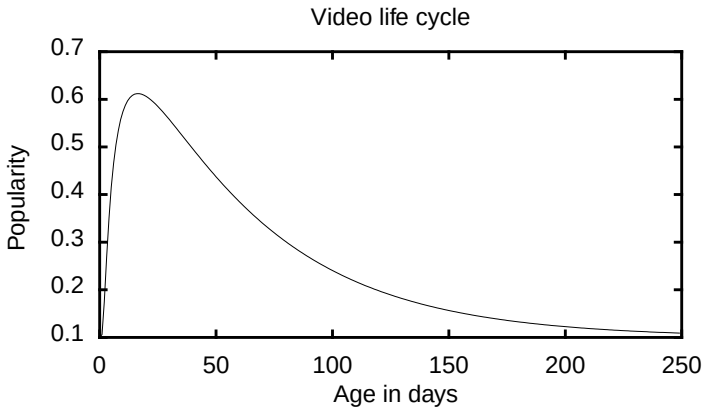


Figure 44: Aging according to *RP(t)*

We found parameters applicable in the case of movie rentals by least square-fitting the function with the movie rental data.

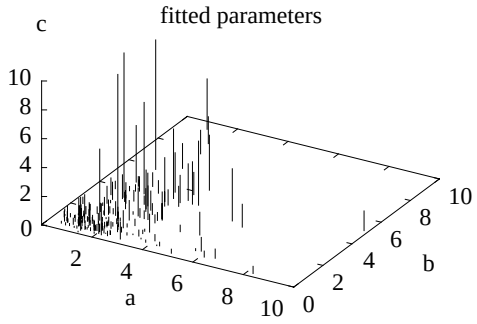


Figure 45: Parameters derived from data

Figure 45 illustrates that the parameters calculated from the least-square fitting with real world data of a video rental store are typically small and show no obvious correlation. The check verifies that the coefficient of correlation for each pair of parameters is tiny (the biggest is 0.01 for a and c, where a can be considered the decline of popularity and c as the remaining popularity).

Because of this observation, we select the parameters for movie modeling independently from each other. For the individual draws that determine the parameters of a newly inserted movie, we use the exponential distribution. That gives us the basic functionality of frequent small and rare big parameters.

6.4.3 Effects of User Population Size

Among the most important observations that are not intuitively clear before they are observed is the vast divergence of user access behavior from the average behavior when user populations are small. The coefficient of variation depicted Figure 41 has illustrated this effect.

Figure 46 demonstrates how our model imitates the effects on the rental probability of a single movie, and also how it recreates the smoothing effects of increasing user population sizes. The life cycle of the example movie and all other movies used for the experiment (initially 150, 1 new movie each day) is defined as presented in Section 6.4.3. Each draw is considered a

request for one movie, the number of hits indicates how many of these draws select the example movie. In contrast to the simulation model that we develop, the number of hits per days was fixed for this experiment.

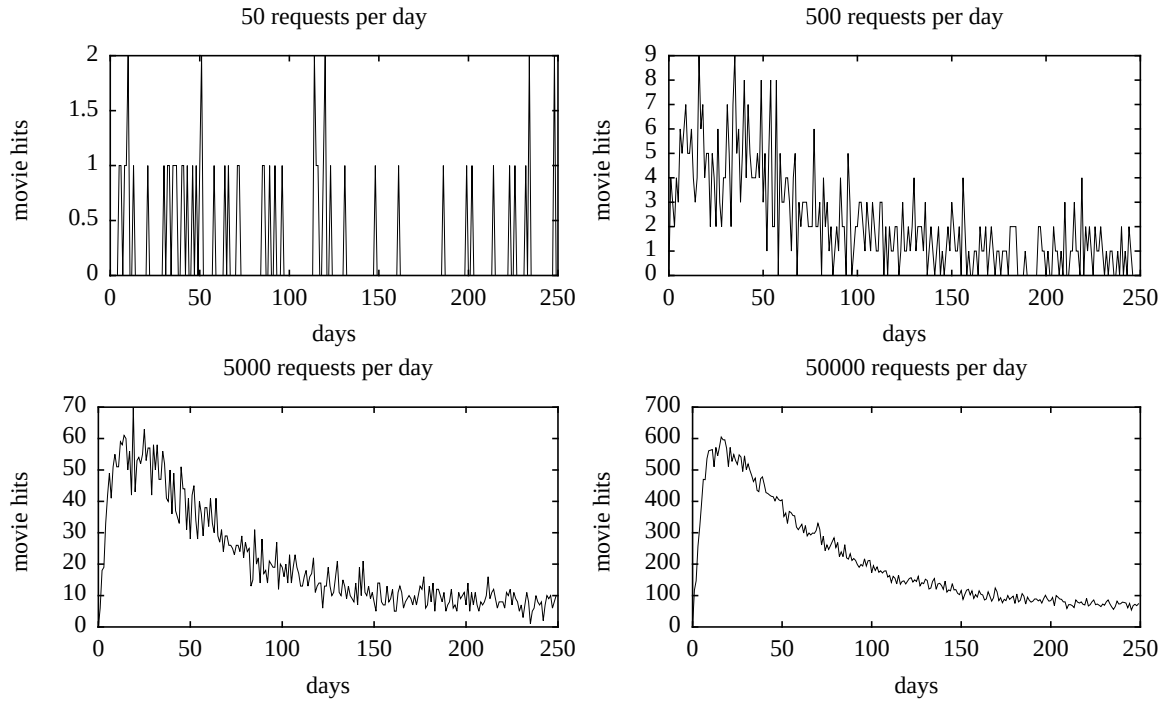


Figure 46: Smoothing effects of growing user populations

Figure 39 implies that caching algorithms that are designed under the assumption that the Zipf distribution provides a worst-case or at least an average-case boundary for movie hit probability may underestimate the number of cache misses in a server at a low level of a distribution hierarchy.

In Figure 47 we show how daily hits according to our model are distributed and compare them to the real-world rental probabilities that we used in Figure 43 before to illustrate the divergence from the Zipf distribution. From 50 draws, we show the highest and the lowest curve. The behavior of our life cycle function yields a more wide-spread hit distribution when compared with the real world data. It is definitely more pessimistic then the Zipf distribution. We observed also that our model will typically generate curves closer to the lower curve, which implies that a distribution of hits over multiple movies is the typical case, while the generation of higher curves, which implies the dominance of a single movie, is a rare occurrence.

Since we have not been able to acquire information about potential user behavior, i.e. the order and frequency of requests to the system, we have no adequate model for this. In our investigation, we have thus used a Poisson distribution, which is memory-less and does not prefer any specific time during the day. We are aware that current television practice differs from such a smooth model of user requests throughout the day - on the other hand, the rental shop data that we have available has shown evidence that financial incentive can convince customers to view movies in the early morning hours. The greater divergence seems to exist between the customer behavior during day-time hours (children) and night-time hours (adults). The next sections deals with the problems of such day-time variations.

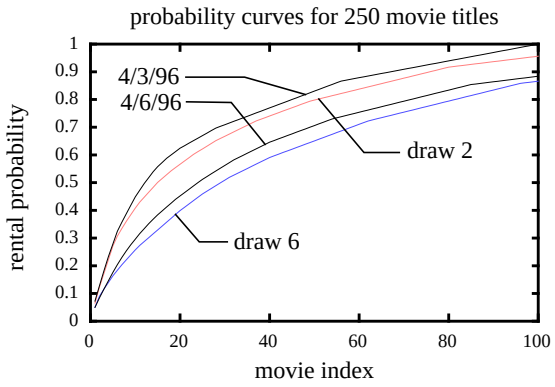


Figure 47: Rental probabilities compared with RP curves

6.5 Day-time Variations

With web caching, depending on the size of the cache space can result in three effects:

- the web cache content stores and replaces continuously and unconditionally, e.g. all contents are replaced once per day, because the space is too small to keep material; items are frequently hit only once
- the same, but the user base is large enough: items are still hit many times
- the web cache is fairly large, and items are removed mainly because their expiry date or their assumed expire date has been reached

With videos, we have a lot more knowledge about their relevance. We have earlier investigated their long-term aging behavior that would warrant a large amount of cache space to be put aside for the most popular movies.

For short-term investigations on the time-scale of a few minutes, as it is used by video server research work, this model would yield a sufficiently exact model. However, a simple Zipf distribution of hits is probably sufficient for such a time scale anyway. Our investigations aim at a larger time scale; several days are intended and the goal is to understand the movement of movie titles among video caches in an environment of several caches to interact. However, our long-term model can not successfully describe an effect that is observed in the real world: depending on their genres, movies' popularity differs with the time of day as well. We know that various TV shows exhibit a different level of attraction depending on the time of day when they are broadcasted. Such day-time variations exist in the TV for various reasons, and many of the schedules have undergone a historical development. For example, comics strips are aimed at children and are on display primarily in the early morning hours, sex and crime at adults and are on display at night. Though there is indication that this assumption would also be true for VoD, we have no appropriate figures to rely on:

- television assumes this user behavior as well, and the programmes are structured according to sociological studies rather than feedback from experiments
- theme channels, which would be sources of information based on their daytime-related popularity, have a very restricted customer base - and providers do not want to surrender figures
- hotel TV is atypical for our scenario

- video rental stores (that were investigated in our older studies) can not provide information about the time of the actual viewing
- laws prevent some movie genres from being broadcast at certain times of day

We can assume that TV schedules are a lot stricter than actual viewing behavior in VoD systems would be; video rental data support such an assumption. Day-time variations are also influence by local law, in some countries such as Germany, certain genres must not be delivered at all times of the day.

In our simulation, we examine the performance of typical caching strategies for complete objects in VoD structures first. These are simple algorithms such as FIFO, LRU, LFU, LRD, IRG. These may not reach the performance limit that can be achieved with more intelligent strategies, especially those for cooperative caching. One indication that cooperative caching is worth the effort can be drawn from intense web caching studies such as [Tewa98] and the Squid architecture; we expect that the advantages of such strategies for videos will be even greater since more meta information about the user response to available content can likely be stored per item than with web caching. The main reason that we make this assumption is that the size of a single data item as well as the overall number of available data items for video on demand movies is such that a large amount of collected meta information can be afforded - which is an important difference in comparison to web caching.

For the moment that only approach that we can follow is the modification of the original long-term model with various kinds of relevant daytime variations. Our intention is to verify that some algorithms (of medium sophistication such as the IRG) loose a lot of their advantage over dumb strategies such as FIFO when daytime variations occur, e.g. in terms of bandwidth consumption between a cache server at the head end and its uplink server.

Due to the lack of a model, we choose some random waveforms with a 24h cycle to modify the long-term probability of our modeled movies. Such waveforms are a sinus wave, sawtooth and skip function with various intensities of affect on the original movie relevance.

6.6 Implementation

This section gives a compressed overview of the features and restrictions of the simulation program that is the basis of the investigation. It presents the abstraction of the delivery path and finally, the resulting class hierarchy. The decisions that are the basis for this design are a compressed form of the arguments for abstraction and simplification given in Section 6.1:

- Since the size of user populations has a major influence on the effects that prevent cache servers from detecting video aging, its model must be realistic. This means that the number and randomness of requests must be kept on the level of a one-to-one simulation.
- The same applies to the number of cache servers that are connected to a higher-level cache server.
- It is assumed that multicast can be exploited.
- It is assumed that control communication experiences no delay except for the delay that is introduced by the distance between two nodes.

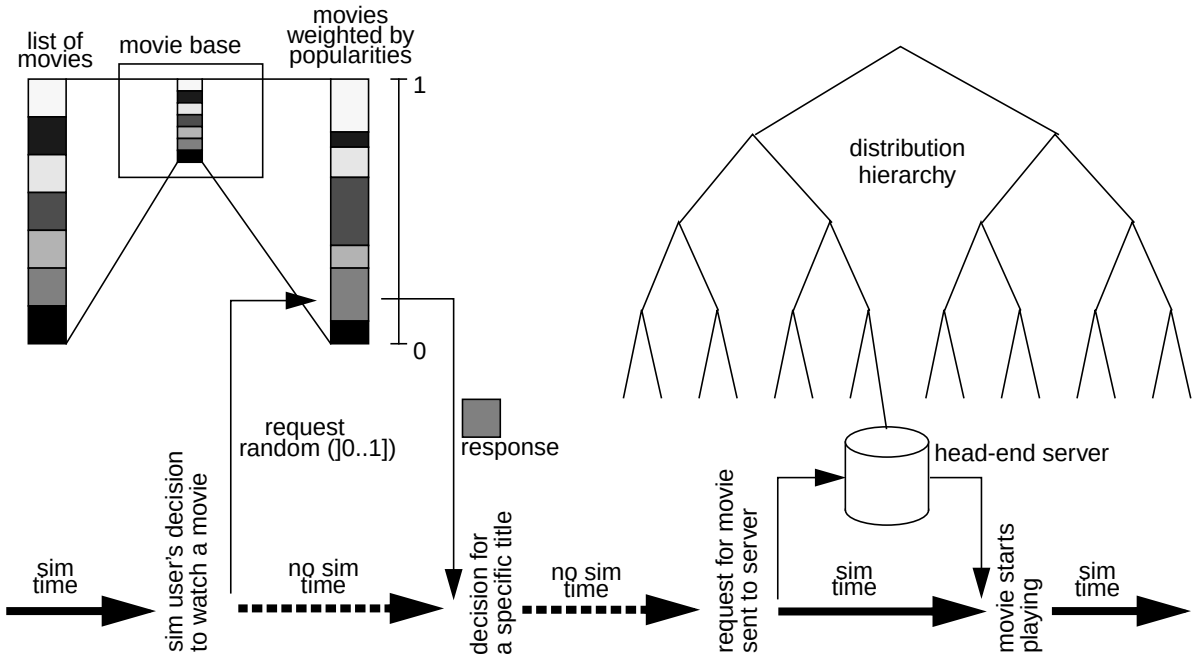


Figure 48: Movie Selection Process

- Cache servers are assumed to implement write-through caching without additional delay to the stream that is transferred from am upstream cache server to an end system. This is due to LC-RTP which provides protocol extensions to RTP and enables it to support end-user delivery and server-to-server file transfer at the same time.
- We assume that data is always streamed at playout rate between servers. Alternative inter-cache communication may require bulk transfer at higher rates. This limitation of the simulation could be removed in the future. To achieve results earlier, this simplifying decision seemed important.
- For pre-distribution among caches we allocate network bandwidth only once since the LC-RTP extensions provide for reliable multicast.

Selection Process

Figure 48 shows the simulation model that we are using. It is built with various applications in mind. It can be attached to a multitude of simulations for distribution models, allows the simple addition of an enhanced user model, and implements correctly the life cycle model that was derived from our recent studies. We have also taken care that the addition of community-specific interests can be supported by adding alternative weighting functions to the movie base.

The list of movies that are available to the system is maintained by a component called movie base. The stored information contains the age and the parameters of the specific movie title. From this information, the current popularity of the movie with respect to the overall set of available movies is computed according to the life cycle function *RP(t)*. Since our current model has a temporal resolution of single days, and new movies are inserted at a rate of one in 12 simulated hours, the movie popularity changes only twice in a simulated day. We make use of this to implement a faster movie selection by computing a second list of the movies

weighted by their popularity. The popularity is normalized so that 1 is the sum of all movie popularity.

A random process, this is a Poisson process until we have more information on same-day user behavior, decides when a user decides to view another movie. A random number in the interval]0..1] is generated and sent to the movie base. This random number identifies uniquely a movie in the list of movies weighted by their current popularity, and the movie is returned to the user as the selection. This movie title is then used in the simulation as the movie that is requested from the head-end server to which the user is connected.

We keep the global probability of one movie to be selected from a large number in line with the probability defined by its life cycle by using a single generator function. This function selects randomly from a common database for all movies. Otherwise, if we had a random selection mechanism built into the user model, probably with an independent number generator in the user itself, bad random number generators could spoil the results.

Due to this approach, a movie has a probability of being selected by a user of the system which is in line with its life cycle function. We assume a true on demand system in which all users are equally informed about the availability of a movie, and the movie is equally advertised towards all users. Since the topology of the video-on-demand system is assumed to be based on physical constraints and not based on user interests, the probability of a movie to be requested is the same for all users. Consequently, the request probability to each movie is same at all head-end servers that serves the same number of users.

The centralized approach has more advantages. A central movie base allows logging of the selected movie and cross-checking for viability of results. Statistics can be kept with the single movie to the end of the simulation rather than evaluating raw data afterwards. The insertion of new movies is more easily modeled if only one database of movies is maintained, and it reflects better the model of a single distributing source for the specific movie. The number of entries in this movie base is increasing throughout the simulation once per interval (e.g. daily), to account for the number of movies that are actually published.

The introduction of intervals is used for another simplification in our model. The rental probability of all movies is updated after an interval that is fixed for one simulation run. In this way, the weighted list of popularities can be re-used for the requests arriving in one interval instead of recalculating the exact probability value for each user request. This simplification saves computing time and seems permissible because the user behavior is not modeled on a single-day scale.

The user model is based on the probability that a user will want to see a (any) movie. It defines the time at which a user will see a movie. We expect to enhance this request for a movie later with certain additional information, like the category of the movie requested. It will also be enhanced towards modeling time of day. Right now, this part of the model is simple and open for enhancement in future versions of the simulation.

Users are modeled as individual entities that initiate the retrieval of a movie. We expect the number of customers of a single video-on-demand server to be approximately constant, with a roughly constant interest in watching movies, i.e. the total number of movies that are retrieved by the customers is fixed. Since there are no restrictions to retrieving any video at any time in a true video-on-demand system, the model does not take any dependencies between various

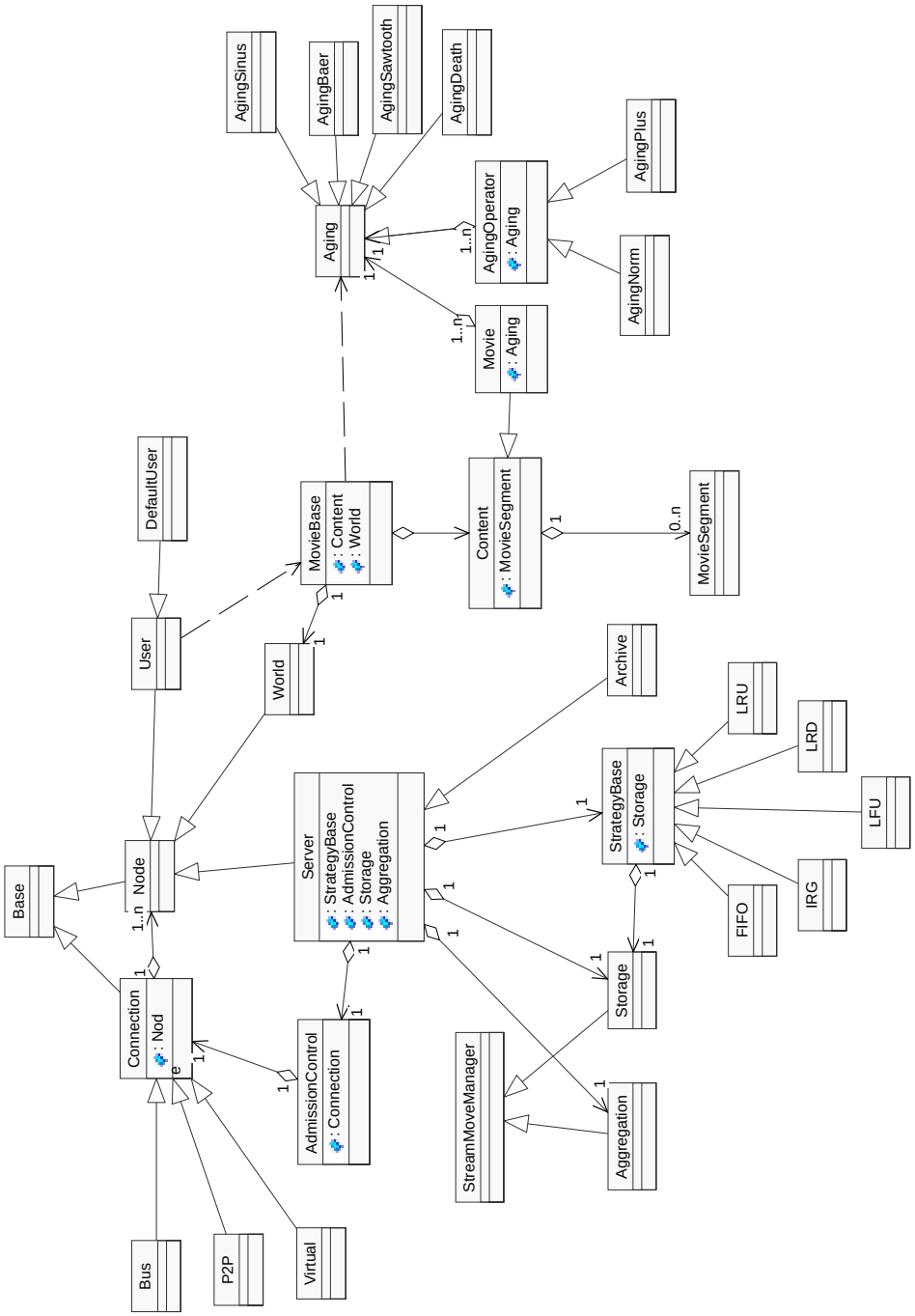


Figure 49: Simulation Class Hierarchy

users into account. The only way for users to exhibit similarities is because of inter-dependencies arising from the handling of their requests by the on-demand system.

Class Diagram

Figure 49 presents the class hierarchy of the simulation. The class hierarchy implements the two main elements that are dictated by the independent modeling of movie life cycles and user behavior. Each of the presented classes has also an instantiating class that represents the class during parsing of our simulation specification language.

The movie life cycle is defined at the creation time of a Movie object and stored in an Aging object that is instantiated at the same time. This separation allows the development of highly complex models for movie life cycles that include long-term variations as well as day-time variations, while movies can be re-used freely. Movies are stored in the MovieBase after their creation. This MovieBase implements the “decision” process for the users and defines the existence of a movie in the provider’s Archive - these two interconnections link the selection model to the distribution model.

The distribution model starts with a Base class, which is inherited by the abstract classes Node and Connection. The child classes of these two abstract classes implement the distribution tree, as nodes and vertices of the system. The child classes of Nodes implement all machines that are involved in a distribution, namely the Archive at the content provider, the Servers and the Users. The Server implements a large part of the functionality of the distribution system, by instantiating one Strategy, one Storage and one Aggregation each. Strategies are varied frequently and control the acceptance of movies into the Storage, or the removal from the Storage. Storages implement the amount of storage space available to the server. Aggregations implement the pass-through feature of the system; they exist because request forwarding time can be exploited by request aggregation. Storages and Aggregations compete for downlink bandwidth as well. The access to this link is controlled by the AdmissionController class (preemption is implemented), which is necessary since streamed transmission requires throughput guarantees.

6.7 Evaluation

We have dealt with the completeness and appropriateness of our model, and with its differences from and enhancements over previous approaches to a achieve a basis for VoD simulation. We have also presented the design of our simulation program. We are aware of a couple of restriction of our model:

- We lack information about request distributions, and thus we use a limited, straight-forward user model. We assume Poisson distribution to remove dependencies.
- We lack information about real-world day-time variations. We assume that such information could only be drawn from field trials that publish their results, probably also from wide-area PayTV NVoD system; neither have published such results.
- There is no information about local behavior of user communities - although this exists for certain. European cooperations in TV productions have shown that the diversity of interests among countries is very hard to overcome.

In spite of these restrictions, we consider our model a step beyond existing ones that we could have used for our investigations. In contrast to many of the earlier studies that have dealt with short-term effects of user requests on servers and networks, this thesis requires a model of the overall effects of caching on a wide-area distribution system. While the older models could without restriction of their accurateness neglect mid- and long-term popularity changes and their effects that become visible in a range of minutes, we can not do this. The caching of video objects that are moved in the network at playout speed takes place on a scale of hours, and the removal decision takes place on a scale of days - the latter is specifically important for small caches that are not able to store the results of all requests that have occurred during a whole day. Due to these strong differences, the basis of the simulation model that is presented in this chapter diverges from the models that have been required for the short-time modeling.

If we would, in comparison, use the older models unquestioningly in our simulations, we would experience a series of problems. For example, the existing short-term models have been time-free, or have been enhanced with a rudimentary notion of popularity change in time. Examples have been given in Section 6.2. With these simple notions of change in time, the popularity changes of movie titles would behave unrealistically, either smooth or chaotic. As an effect of overly smooth modeling, the simulation would indicate required network resources far below the actually required values. With chaotic behaviors, the contrary applies. We believe that through the separation into two models, the predictable ‘aging’ quality and the unpredictable user ‘choice’, we achieve a behavior that comes close to reality. We concede that a clustering of interests has not been investigated yet; if such a clustering existed, it would certainly indicate that the installation of a cache server (of unspecific size) for each cluster would be appropriate.

Daytime variations may have a strong effect on popularity at a certain time. Since we have not had access to any material that would allow the modeling of such popularity variations, we are applying selected modifier functions to the popularity to investigate possible effects.

7. Simulation Results

This chapter of the thesis investigates the effects of distribution mechanisms and removal strategies that have been mentioned in the previous chapters. It addresses only a strictly limited number of the questions that could be investigated in continuation of the previous chapters and the facilities of the simulation program.

Although simulations are typically used to model systems in great detail, the investigations undertaken here aim at conceptual differences between strategies. For various reasons, extensive studies of the detailed results are not considered a good idea. One reason is that although the long-term popularity development of movie titles is investigated closely in this thesis, the user model that is available at this time is very limited due to a lack of publically available information; the broadcasters who collect this information consider it confidental. Another reason is that the number of caching strategies that should be applied for a detailed investigation is large, with a growing number of caching strategies that spring into existance in coordinated web caching.

Still, even when the investigations are limited to the examiniation of concepts, a vast number of aspects and their interaction can and should be investigated in addition to the conclusions drawn from this chapter.

The following simulations were modelled with a specific set of parameters in mind. Additional parameter sets would have been interesting, but in spite of intensive tuning and parallel execution, an investigation would take up to one week, which makes this impractical. The bandwidth required for streaming of one movie is supposed to requires 1.5 Mbit/s of the raw network bandwidth, which reminds of MPEG-1, while the raw network bandwidth available is chosen as 155 Mbit/s and 622 Mbit/s. This ignores the protocol overhead in ATM, or the effects or VBR MPEG streams. Network delay is constant for each link, and mainly intended for examinations of the original cache level of movie that are received by the users. Although possible, overload situations at the servers are not considered in the simulation, except for exhausted downlinks or filled caches. The results of Section 7.1 imply that user population sizes of 5000-10000 concurrent users are applicable, which can be served by large-scale video server products that exist today.

Due to the presented analysis, we expected an overwhelming influence of the distribution mechanism chosen, and a limited influence of the removal strategy. The simulation, however, dictates different conclusions.

7.1 Effects of increasing user populations

Analytical investigation indicated that proper placement and *gleaning* would be highly beneficial. However, the placement that is found by the analytical model requires a strong cooperation between caches in the hierarchy, to prevent cache servers close to the customer (we say: at the lower levels) from caching the most popular movies. For practical reasons, it is noted that lower level cache servers have, on the contrary, a high inventive to prefer the storage of the most popular titles. Doing so reduces the dependency on available uplink resources for the

most popular titles and results in the highest average user satisfaction. We envision less cooperation in this simulation and demonstrate the beneficial effects of gleaning nonetheless.

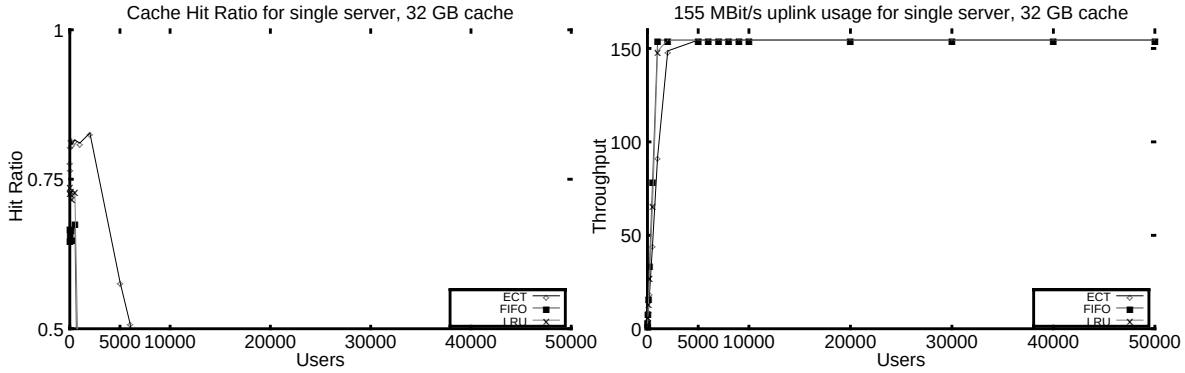
We have stated in the previous chapters that user populations have a major impact on the capability of caching strategies to determine, without additional information, the most popular contents. These contents should then be cached. However, the implemented local strategy, which was not considered in detail up to now, plays a major role in the efficiency of a cache; at least where the most popular titles are concerned. Section 6.3 indicates that there is no means for efficiently determining medium popular contents. Figure 50 is intended to demonstrate two key issues considering caching of the most popular titles:

- conditional overwrite strategies can be highly efficient compared to unconditional overwrite strategies
- the limited uplink bandwidth is quickly exhausted and the performance degrades immediately when storage space is limited and does not allow the strategy to store a sufficiently large working set of titles

The setup of the conditional overwrite test consists of a single server with 155 MBit/s uplink capacity and varying cache sizes. The caching strategies that are shown in the Figure 50 are (additional removal strategies are shown in the appendix):

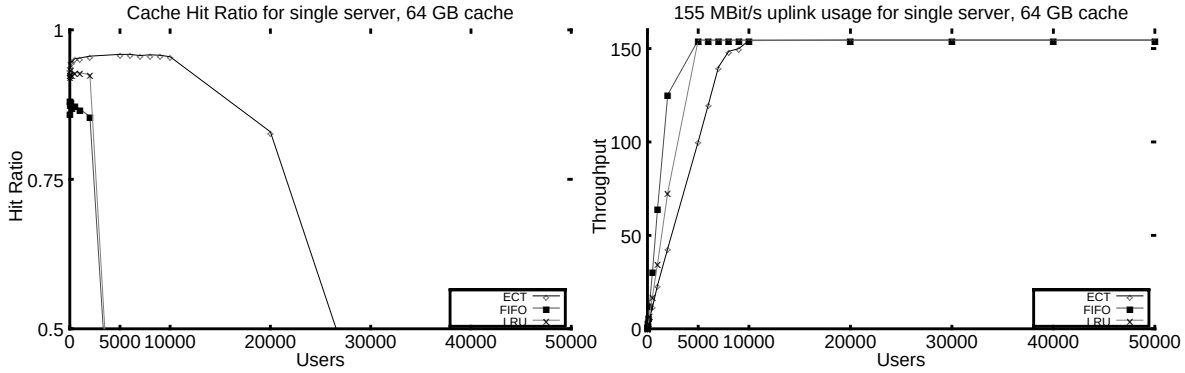
- FIFO: The first-in-first-out strategy.
- LRU: The least recently used strategy. In LRU, each hit to a cached element makes it the least probable element for removal, new elements are always cached and replace the most probable element for removal from the cache.
- ECT: In the original inter-reference gap strategy all requests are counted and for each cached element, a history is maintained. An entry in the history is a distance between two consecutive hits to the stored element, expressed in intermediate requests for other elements. When a new content is requested, the content with the largest average gap is replaced. ECT is an IRG-variation that keeps the hit array for an element even after it has been removed. It is a conditional replacement strategy: first time selections are always cached, but after being removed once, the history for the entry is not deleted and taken into account in all consecutive caching decisions. Like in the original IRG approach, the size of history for each entry is 8. The short-hand ECT is derived from **e**ternal history, **c**onditional replacement and **t**emporal gap size.

Before the initial simulations, we expected that LRU would be a good reference strategy. We learned that this is only true for strategies that cache items unconditionally. Among those strategies, its performance can be considered a reference. Its efficiency is better for small caches that hold only the most popular titles. These most popular titles are those that are hit more often than any possible sequence of different other titles that is long enough to take up the entire cache space. The observed popularity distribution which represents the popularity curve is falling exponentially, while the number of consecutive misses that a title can take without being removed from the cache is strictly linear with the cache size. This implies that LRU will be efficient in comparison to more complex technique as long as the caches are small with respect to the number of available items. The results of for several unconditional overwrite



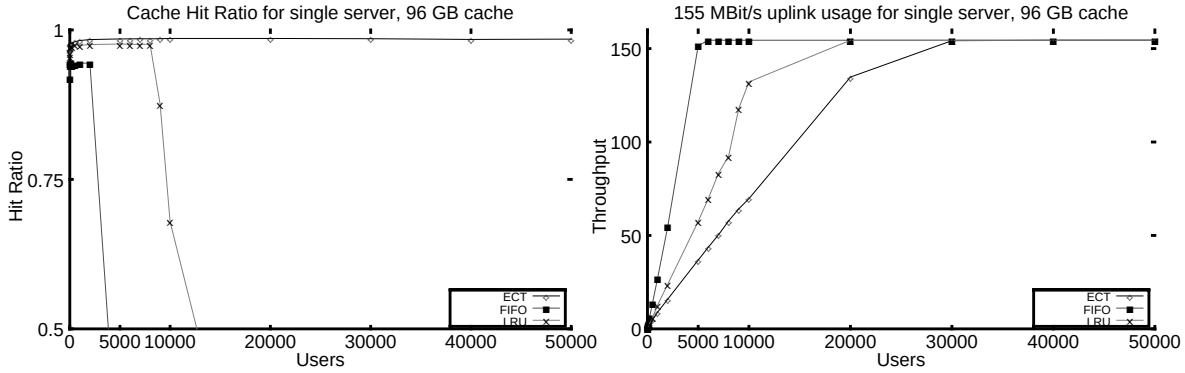
(a)

(a')



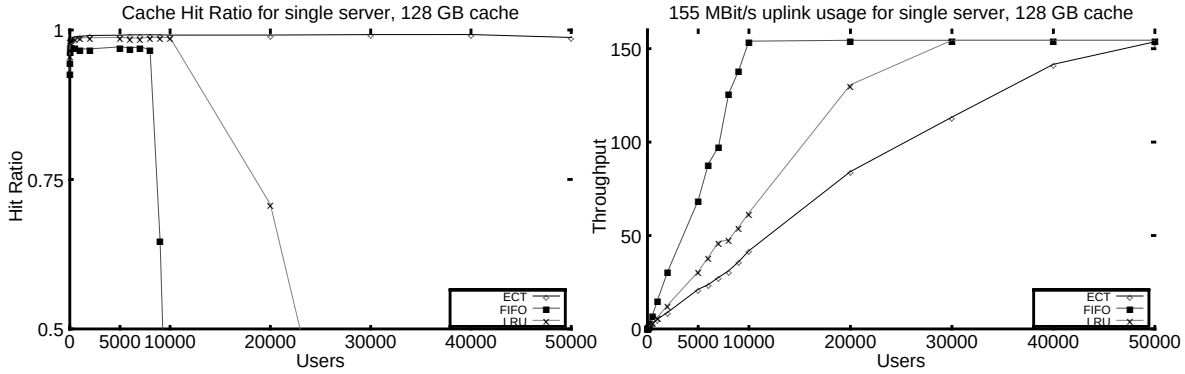
(b)

(b')



(c)

(c')



(d)

(d')

Figure 50: Effects of caching strategies on user hit rates and throughput

strategies are compared with LRU in the Appendix 10.5. They support this consideration. The comparison with conditional strategies is more interesting.

The potential of conditional overwrite strategies is demonstrated in the hit ratio graphs Figures 50 (a)-(d); compared with unconditional overwrite strategies, the ECT strategy can support a multitude of users with the same cache size.

These results may be typical for simulated caching strategies of content that ages slowly with respect to the number of requests to the cache, but important differences to other applications of caching exist due to the application scenario. With web caching, for instance, the information that would be required to store ECT information would exceed the size of the average content quickly and thus, would be an unacceptable waste of storage space. For movies that have the size of some gigabytes, the caching information takes only a negligable amount of space. For applications such as paging, swapping or CPU caches, the option of conditional overwrite does not exist since local presence of the data in the cache is required for an operational system operation. Distributed databases need completely different considerations due to their consistency requirement.

Without consideration to the bandwidth consumption on the uplink, the hit ratio figures 50 (a)-(d) could be interpreted as an indication that a simple strategy can be used without penalty in conjunction with an affordable amount of disk space (e.g. 96 GB) to serve a realistic number of users (e.g. 5000) by installing a single cache. The Figures 50 (a')-(d') show the usage of the uplink between the cache and the library server. They demonstrate that there is in fact a penalty. While the demonstration of the advantages of the conditional overwrite strategy are hardly as convincing as the hit ratio above, the better resource use is clearly visible.

The fact that it is less clear in these graphs than in the hit ratio graphs is an inherent effect of the simulation design: users have a limited patience (5 minutes) and cancel the movie request after this time and thus, they waste only 4.5% percent of a movie length (90 minutes) in the waiting state, then they cancel their request and retry up to 3 times with a newly drawn title. If the uplink is not exhausted, all of these requested titles that are not cached will be served by the library server directly. As soon as the uplink is exhausted, requests will be either refused by the server or cancelled by the user, whichever happens first.

Based on the results of these first simulations, we present the following simulations exclusively with the conditional overwrite strategy ECT, which achieve considerably better results than the unconditional strategies. Results with other strategies can be found in the appendix.

Starvation effect

We detected that a cache starvation phenomenon that is not expected with a conditional overwrite strategy can still impede the performance of a cache server under rare conditions. Initially we noticed this due to a simulation bug that was subsequently fixed. It can occur under two conditions that involve conditional overwrite strategies on relatively small cache servers.

One condition requires an extreme overload of the uplink. Due to this overload, the cache will rarely receive the movie that it has decided to store locally. When the caching decision is finally honored, the popularity of the movie is decreasing again already.

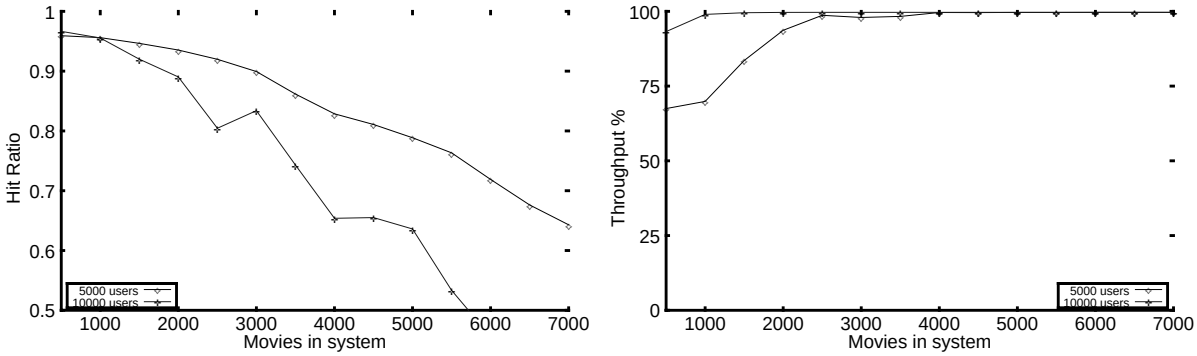
The other condition, which assumes that the cache does not deliver streams from a movie copy any more when it is scheduled for replacement by a requested, but not yet incoming movie. Under this additional condition, the uplink overload needs not be as bad as for the previous condition, yet its effect is worse. The cache is quickly deprived of movies that are still served, yet the probability that an incoming movie from the next level server is a title that is chosen for caching is not predictable. It is mandatory for a caching mechanism to allow the cache server either to refuse user requests instead of forwarding them when they can not be served, or to be able to inform the next level server whether a movie is requested for caching as well as for forwarding.

We modified our strategies to refuse user requests to contents that are not selected for caching. The chosen threshold is reched when requests are pending that would result in a replacement of 50% of the cache size.

Increasing number of movies

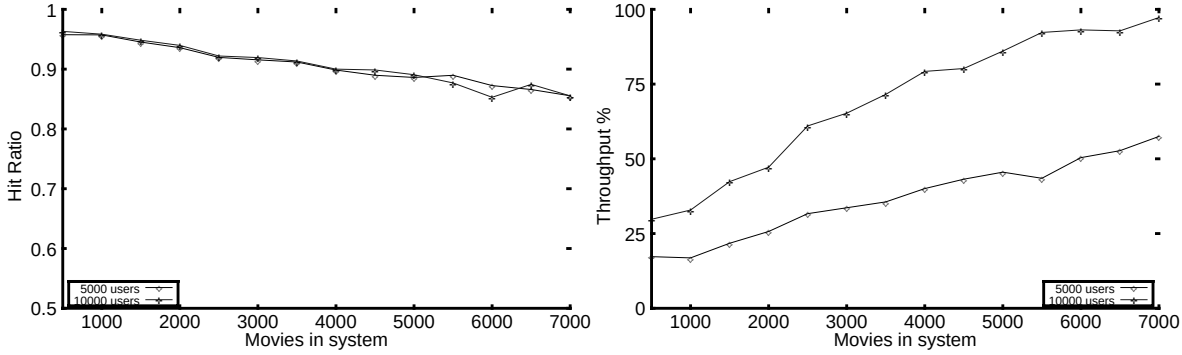
Our previous investigations were made with a changing set of 500 movie titles that were concurrently available for selection by the user. Each title ages according to its randomly generated parameters and is removed from the set of available titles after 500 days.

Taking the fact into account that movies have an individual residual popularity (Section 6.4), we assume that the hit ratio of a cache is falling when the number of concurrently available titles increases in an otherwise unmodified scenario. This is also predicted by the Zipf distribution. The actual effects of this variation can be seen in Figures 51 and 52, which show the results of a presentation with a fixed cache size (64 GB) and two different user population sizes (5000 and 10000). We increase the number of available movies gradually from 500 to 7000 and examine the effects on a cache server that caches and discards titles from the cache according to ECT and that has an exclusive (i.e. non-shared) uplink. We notice a strong influence of the number of titles on the efficiency of the system.



(a) (b)

Figure 51: Decrease in hit ratio (a) and increase in uplink usage (b) with increasing number of available movies, uplink capacity 155 MBit/s



(a) (b)

Figure 52: Decrease in hit ratio (a) and increase in uplink usage (b) with increasing number of available movies, uplink capacity 622 MBit/s

Figure 51 illustrates that the hit ratio decreases quickly with an increasing number of accessible movies for both 5000 and 10000 users. We notice that the decrease of the hit ratio is considerably more moderate when the uplink is 622 Mbit/s rather than 155 MBit/s wide (Figure 52).

Obviously, the exhausted uplink results in the unexpectedly steep decrease in hit ratio. This happens in spite of the protection that we use in reponse to the starvation effect. The log files indicate the efficiency reduction would be much worse without this protection.

We conclude that popularity of movies should be taken into account when titles are exchanged between cache server. A high average success rate (i.e. number of users who receive their requested title out of all requested titles) can be achieved in several ways. One approach requires that the higher level cache server stores the frequency of requests to movies and that it preferes the transmission of more popular titles; to prevent starvation of lower level cache servers, these should be able to indicate requests for movies that they intend to cache. Alternatively, lower level cache servers could derive this decision on their own, probably in conjunction with hints (Section 7.4), and stop requesting movies when they detect that the uplink its overloaded; this approach is prone to misbehaving neighbours.

On the other hand, these approaches may reduce the attractiveness of video-on-demand as a service that delivers rare movies to the user.

7.2 The Bandwidth Effect of Gleaning

To demonstrate the advantageous effect of Gleaning, we consider two levels of caching, where cache servers at the first level serve 5000 users each and apply ECT. These first level caches share an uplink that connects them to the second level cache.

The two scenarios shown in Figures 53 (a) and (b) differ in two simulation parameters. For the first scenario, we use a 96 GB cache and a 155 MBit/s shared uplink. For the second scenario, we use only a 64 GB cache and a 622 MBit/s shared uplink. The figures show the refusal probability that is experienced at the client rather than any hit ratio or throughput values.

With the scenario of Figure 53 (a), we achieve a 98% cache hit ratio that accounts for the small number of requests to popular movies that is forwarded to the second level server. We

know that the cache serves most requests directly and that the requests to titles that are not cached are spread over the set of uncached titles. Of course, the number of these requests grows with the number of first level cache servers that are served by the second level cache server, and the shared uplink between first and second level is exhausted at some time, resulting in service refusals. With a refusal probability below 1%, a connection of 10 first level caches to one second level cache server seems appropriate.

It is interesting that we observe only small differences between unicast delivery, batching with a 5 minute window and gleaning. This is obviously due to the rareness of consecutive cache misses at the first and cache hits at the second level. We notice that gleaning performs slightly better in this scenario than unicast and batching.

The results change radically when we examine the scenario of Figure 53 (b), which allows only a 94% cache hit ratio at the first level cache. This change in cache size provides a higher probability that movies are requested in shorter sequence at the second level cache server. We assume that this results in a better performance of batching and gleaning. The investigation shows that this assumption is only partly correct. Batching performs even worse than unicast; the explanation is the increased delay in delivering the videos, which increases slightly the probability of cancellation due to an impatient user. The application of gleaning, on the other hand, demonstrates that it can operate efficienctly in this environment; an increase in refusal probabilities can not be seen in Figure 53 (b).

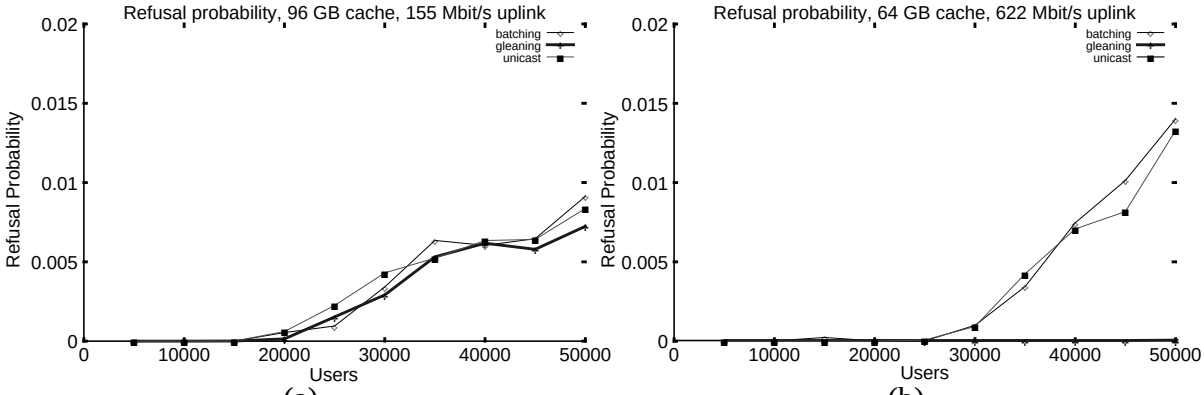


Figure 53: Refusal probabilities depending on user hit rates

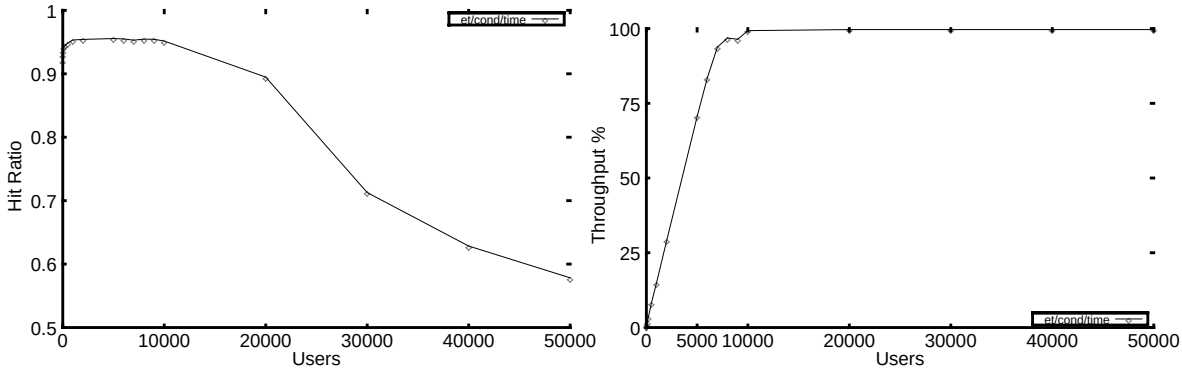
We conclude that gleaning can be highly efficient in a de-centralized setup, even though the optimal placement of movies in the distribution hierarchy is not achieved. To operate more efficiently than simple distribution mechanisms, a multicast network and a considerable number of connected caches is needed, where the set of movies that is cached at the first level must be similar for each cache.

7.3 The Hit Rate Effect of Daytime Variations

For a conclusion of single server effects, the effects of strongly varying popularities for all titles throughout the day were simulated. This is intended to understand the effects of the real-world phenomenon that different movie genres are preferred at different times of the day. We wanted to understand in which way the expected replacement of the cache content affects the performance of the caching strategies.

The assumed result was earlier that the frequent exchange of movies between levels of the hierarchy would severely affect the overall bandwidth requirements. This assumption changed subsequently with the observation that techniques such as gleaning can strongly reduce bandwidth requirements. The simulation results support the opinion that there is no penalty in daytime variations, but they indicate also that the reasons are different.

The realism of our daytime variation simulation is limited by the lack of a proven model of real-world user behaviour. Several potential sources have been considered and discarded due to the lack of realism. Broadcast television uses a self-fulfilling model by assigning times of the day to certain genres. Rental stores do not collect data on the actual viewing times of their customers. Hotel television has a very restricted programme. Pay-TV channels like Premiere World in Germany would be an appropriate source of information, but right now this information is not released.



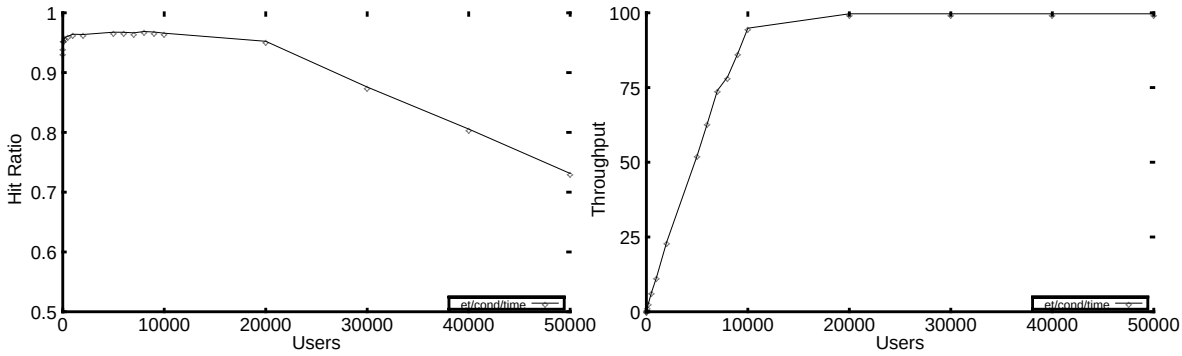
(a) (b)

Figure 54: Development of hit ratio (a) and of uplink usage (b) with increasing number of users, daytime variations with random peeks

To examine the basic properties of daytime variations, we decided to apply additional variations with a 24 hours cyclic behaviour to the long-term popularity curves of our movie model. We start with a sinus function that is applied to 50% of the long-term popularity value at each time. Within a 24 hours period, the popularity of a title varies between 50% and 150% of the popularity that is defined by the long-term model. For Figure 54, the daily popularity peak of each titles is chosen independently throughout the day. To save computing time, the popularities are not recomputed for each selection that is made by a user, but only every 10 minutes (simulated minutes).

It is probably more realistic than total freedom of choice that users prefer certain genres at certains days of time. This would result in concentrated requests on movies in a cyclic manner. Some titles are reaching peek popularity in the morning, and some in the evening. To account for this, we have added simulations that create two daily hot spots. A movie would have its daily peek popularity either close the one of the other hot spot. The following simulation

locates these hot spots at 14:00 and 20:00 hours. The results can be seen in Figure 55. Simulation results for other removal strategies and less incisive parameters are shown in Section 10.5.



(a) (b)

Figure 55: Development of hit ratio (a) and of uplink usage (b) with increasing number of users, daytime variations with two hot spots peeks

By themselves, these figures are no more but another demonstration of a conditional overwrite strategy that delivers good hit rates in spite of an exhausted uplink. When compared with each other and with Figure 50 (b), which shows results without daytime variations, the hit rate increases when the variations are intensified. In spite of changes to the content that is kept in the cache, the “working set” of most popular movies, the hit ratio is increasing when the overall number of movies in the system is kept constant. This happens in spite of an overloaded uplink - it is typically exhausted from cache misses (although the refusal probability is below 1%).

We conclude that the constant load on the uplink for delivering movies from higher level caches renders the rotation of a complete cache content irrelevant. The assumption of a high-throughput link between the first level cache server and the second level cache server, which is the basis of an on-demand access to less popular titles, results in an unnoticable exchange of movies between the levels when the popularity changes. The decision to model popularity changes only by 66% instead of 100% of the highest value results in movements of only one level in the typical case.

7.4 The Hit Rate Effect of Hints

Hint-based caching has been proposed for web caches by Tewari. It works by an exchange of request statistics between neighbouring cache servers, which consider this additional information in their own caching decisions. It is shown that the information exchange between caches improves the quality of caching decision in individual caches considerably. Among the advantages of hint-based caching over centralized decisions are its resistance to connection loss and the ability to account for regional differences in preference by considering the information that is received from neighbouring caches only with a predefined weight.

In the scenario of this thesis, the effectiveness of hints is unclear. On the one hand we expect quality enhancements in our scenario which considers exclusively large objects that are nevertheless stored completely in the cache servers. On the other hand, the ECT strategy is able to collect information for all available titles, and the storage requirement is small enough in com-

parison to the content size to actually keep this data. To validate the effects of hints that is used to augment an ECT strategy we simulate a system that receives hints from 1, 2, 3 and 4 caches in the neighourhood, respectively, that have the same size and user population as the examined cache server. We want to find out how important the faster update of ECT with hints is in comparison to an ECT strategy that relies only on local information.

We observe that the information from the caches in the neighbourhood influences mainly the quality of decisions for small user communities when the unweighted memory of the ECT strategy is big. The number of titles is very small compared to the amount of web content, and the number of items that can be cached is extremely small in comparison to the number of items that a web cache can hold. The popularity of the movies changes slowly, while the number of hits to titles are considerable. This combination allows the ECT to make good decisions even if its memory holds the information of a whole day. Thus we expect that the quality of the caching decisions depends primarily on a good relation between the memory size and the user population size than on the support by hints. Figure 56 supports these considerations; the figures show two example scenarios with 2000 movies, daytime variations with two peeks, a cache size of 32 GB and an uplink bandwidth of 155 MBit/s (note that the scale on the X-axis is logarithmical in contrast to the other simulation results). The chosen values are intended to produce results with quickly, with small user populations. In the figures, the number of hints is constant for each curve, rather than relative to the number of users that are attached to a cache. The effect of these hints is reduced when the number of users increases, but in the given scenario, the effect is not totally lost until the starvation effect breaks the strategy.

It is also important to note that the larger log of the ECT strategy in Figure 56 (b) has nearly no effect for small user populations. For larger populations, it prevents misjudgement of the contents' popularity, and the hints continue to improve the strategy.

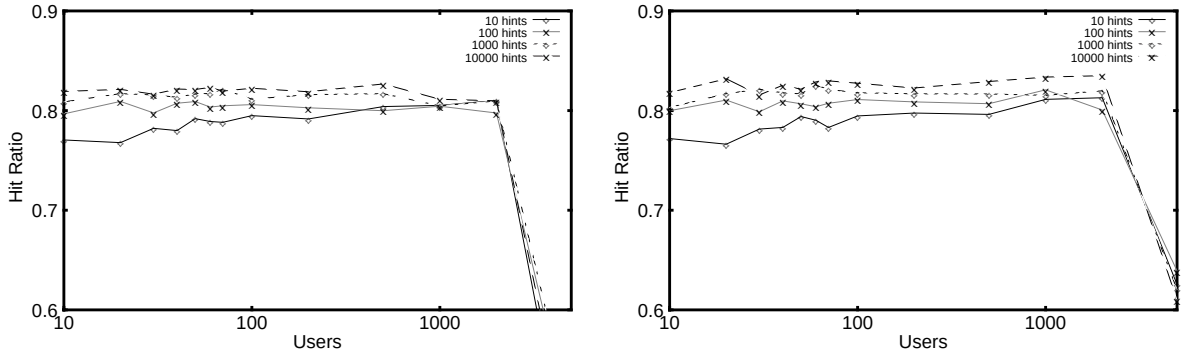


Figure 56: HitRatio development with growing number of users under with Hints and ECT. ECT log size per movie is 8 (a) and 64 (b), respectively.

The experiments were also performed without daytime variations. The results are similar, although the effect of the hints is less dominant. The reasons for the reduced relevance of the hints is the slower change in movie popularity. The slower change allows even the log entries in large logs to remain relevant for a decision, while this is not the case for the shown simulations with daytime variations.

7.5 The Strategy Degradation with Multilevel Clients

If the cache servers at the first level demonstrate any efficiency at all, a 2nd level cache server without directly attached clients will not experience a distribution of the hits that can be approximated by a Zipf distribution. For web caching, this has been documented by Marshall and Roadknight ([MaRo98]). We observe the same effect in our simulation but we have still achieved hit rates around 75% for second level caches with the same capacity as their attached first level caches.

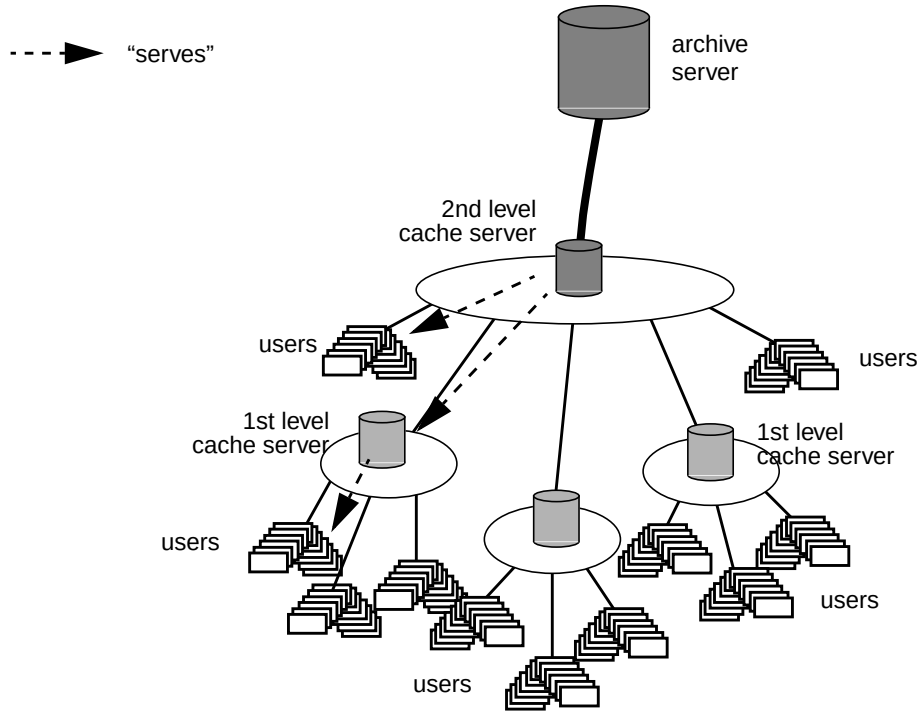


Figure 57: Multilevel clients

Clients sets that retrieve the same content from various levels of the hierarchy are affecting the interaction of independent hierarchical caches adversely. Figure 57 shows this situation which we call "multilevel clients": the cache server of the second level serves clients directly as well as through cache servers of the first level.

In such a scenario, the problem of the purely hierarchical interconnection is the tendency of higher level servers to store primarily the contents that are also stored at the lower level. This is intuitively clear since the number of users that they are serving is considerable larger than the number of lower level servers that are served. Thus, the Zipf distribution of observed requests explains that the movies with average hit rates (that are not stored in the lowest level servers) do not sufficiently affect the higher level servers to store mainly these movies of average hit rates. We can assume that top popularity titles and average popularity titles are stored in the higher level servers, in spite of potential variations in the popularity order that is perceived by the higher level server. The only option that is allowed by a purely hierarchical distribution infrastructure (which should be assumed as a beginning of Internet VoD, just as the Usenet vinetree distribution system started the internet expansion) is to expand the number of titles that are stored at higher level; on the contrary it is pointless to make low level servers large.

This can probably be solved by leaving the hierarchical assumption and by allowing movies which are stored at lower levels to be retrieved from their higher level servers. A simpler solution is the reservation of parts of the larger second level cache for serving the directly attached clients.

7.6 Simulation Wrap-Up

The simulation program that was used for this chapter was developed to reflect many feedback aspects of video delivery that can not be considered in analyses. Several caching strategies were introduced in alternative versions to investigate the quality of their removal decisions. The possible observations included delays, refusal and cancellations at the client side, statistics for all or single network connections or servers, and statistics of all or single movies. While this chapter demonstrates only a tiny portion of the investigations that were made, most were necessary for the understanding of the interdependance of components in the video delivery, but not relevant for the demonstrated results. The simulations that were performed for this chapter led to new insights that were at some points not in line with the expected results. Specifically, the results demand the user of conditional overwrite strategies and confirm the proposal of using hints. It is shown that a starvation effect must be considered when caching strategy and distribution system are designed, and it confirms that very small caches will actually be able to respond to a large number of requests, even though caches that serve only small user populations can gain strongly from long-term observations and from additional information to achieve a good hit ratio.

Use conditional overwrite strategies

Conditional overwrite strategies can be applied in streaming media delivery because long-distance video streaming from a higher level cache server to the client is technically possible. An application of conditional overwrite strategies allows caches to operate at a much higher level of efficiency than typical, unconditional overwrite strategies that are required for many other applications of caching.

Use hints

Hints allow faster reactions to changes in the movie popularities, and allow better adaptation to daytime variations of movies' popularity. Unless there are well-known regional differences between co-located user communities, hints should always be exchanged between their cache servers to increase the performance of the selected removal strategy.

Network requirements

The simulations were configured with MPEG-1 content and streaming-capable backbone networks in mind. For a real-world application of commercial video-on-demand, MPEG-2 or an alternative format may be more appropriate, but MPEG-1 seemed applicable for an alternative to the video rental stores that is envisioned in this thesis. The main influence of an MPEG-2 scenario will be the increased danger of the starvation effect. A real-world strategy must be able to suppress this effect more aggressively than the simulated strategies, e.g. by the cancellation of transmissions at the higher level cache servers.

Small caches achieve good hit rates

In spite of the existing 80/20 rule-of-thumb that is applied in estimations of cache efficiency, the experiences with web caches that store vast numbers of pages due to a single, unrepeated request made me wonder whether the life-cycle model and the randomly generated user requests would actually stay within the bounds of this rule-of-thumb.

It did, and more importantly neither the long-term nor the day-time variations in user popularity had any relevant effect on the hit ratio.

Future work

The simulation should be extended in the future to cover additional aspects. Among these is the requirements to choose a metric for measuring the quality of a cache filling. This would allow to investigate the effects of client that connect to cache servers at various levels in the distribution hierarchy.

Furthermore it is an important issue to simplify the user model further, as a precondition for the simulation of larger distribution trees in acceptable time. Such an improvement would then allow to model non-hierarchical topologies as well. The existance of cross-traffic and the communication among peer will be worth an investigation. We are currently working on a project that would benefit from information about caching among peers. The first new topology that must be considered in this case concerns the support of several root servers. Assuming that wide-area distribution of video-on-demand would actually develop in the Internet, competition between several providers would be seen. In this case, a cache would be part of several logical trees, and must arrange its decisions accordingly.

Another issue is the need to model a mixed workload, including for example movies and news clips, or even web traffic. This is necessary because it is conceivable that an unconditional removal strategy should be applied to news clips with their small size and short life-cycles.

8. Conclusion

The goal of the thesis was the investigation of a caching-oriented infrastructure for wide-area video-on-demand without central control. This investigation has been restricted to a concise data distribution system, while data management aspects, stream synchronization and resource reservation have been left out. These are considered connected but independently solvable research topics that are also under investigation. Storage subsystems have been examined only peripherally as well.

At the beginning of the thesis, earlier work on VoD distribution systems has been structured and evaluated. One conclusion of this evaluation was that single-server research has considered primarily the short-term effects of user requests on the video server and on the distribution systems. While this is relevant to the servers in a wide-area distribution system, it was considered possible that traffic would be more strongly affected by long-term traffic variations when the distribution in the backbone network applies caching. Two examinations contributed to this investigation. On the one hand, statistical data of the popularity of rental movies were analyzed, and an aging model for movies was created. On the other hand, the Zipf distribution, which is frequently employed for user request modeling in VoD was analyzed. It was demonstrated that although it describes *observed hit rates* to an available set of movies appropriately, it does not provide sufficient information for a realistic model of these hit rates of a longer time span.

A structuring of existing caching techniques was also performed in order to understand the basic properties that make up a strategy. This investigation allowed the separation of the caching mechanisms into the two main elements: the *distribution mechanisms* and the *replacement strategy.* Although they impose requirements on each other to operate efficiently, and although the effects of the combination is experienced by end users in terms of delay and refusal in both cases, each can be exchanged individually. The replacement strategy influences whether retrieval actions must be performed by a server; more efficient replacement strategies achieve a higher hit ratio and reduce the number of streams need to be received at the cache. The distribution mechanism influences the efficiency of answering responses to retrieval actions. More efficient distribution mechanisms reduce the number of concurrent streams that need to be transmitted from a cache or library server.

The existing ideas for VoD distribution mechanisms were examined and in particular the stream tapping/patching idea was found as a candidate that can be integrated into a cache-based distribution system. With the so-called λ*-patching*, an independent server is giving a practical means of identifying the optimal time before sending a full-length rather than a partial stream. While patching promises large bandwidth savings in an analytical comparison, a combination with caching reduces bandwidth requirements further. The analysis reveals that the technique is not only applicable for the decentralized architecture that is developed in this thesis. It proposes even better efficiency for a large-scale distribution system that can position content in the ideal position in a hierarchical system due to a central controlling instance. Since it diverges from the target of the thesis, this consideration was not continued in detail. The approach that is considered for the decentralized case combines λ-patching with caching and is called *gleaning*.

After the presentation of the potential for technical viability, an altogether different issue that could prevent the implementation of a wide-area distribution system was addressed. This concerned the copyright problems which are inherent to a system that stores content in multiple copies on untrusted hosts. This thesis does not refer to security issues that are independent of video delivery. Security issues that protect the video stream, cache servers or the hosts are closely investigated in various other research areas and have been considered out of scope for this thesis. Results of such work can be applied to the video distribution system. With copyright protection, the application of existing work to a de-centralized distribution architecture was not possible. To make a decentralized infrastructure viable, an copyright protection architecture named *remark* was developed that implements the possibility to identify copyright violators by error insertion. The approach is able to identify cheating cache servers and cheating end-users indenpendently by repetetive steps of error insertion and error correction. The approach requires the direct delivery of a very low bandwidth sub-stream from the content provider to the end user. This sub-stream is necessary to extract an enjoyable video quality from the delivered content. This allows the content owner to enforce a download notification for each request that is made to a video. The *remark* architecture has the potential of supporting personalized marks that are not visible as errors in the video stream, which was not further pursued in this thesis. The variation requires a detailed analysis of the video stream to prepare stream variations, but this needs not be done in real-time.

With technical viability and legal problems addressed, the issue of backward-compatible protocols was investigated. VoD field trials have resulted in numerous protocol implementations, many of which are proprietary. Typical Internet development, however, has demanded that success for new protocols depends on its operation with the currently existing Internet infrastructure and protocols. With this in mind, a protocol suite was designed that is compatible with established standards and products, works well with proxy caching, supports segmented streams that are required by patching-related techniques, and allows reliable transfer of content into caches servers. The resulting protocol suite consists of *LC-RTP*, *LC-RTCP*, RTSP and SDP. The latter two are standard control and description protocols that allow distribution systems that include abilities such as proxy caching and re-direction (RTSP) and protocol support for segmented transmission of content (SDP). LC-RTP and LC-RTCP are variations of the frequently applied streaming protocol pair RTP/RTCP that is implemented in many products and research prototypes for streaming media. Our variations allow interoperation with existing RTP-compliant clients, they are reliable to achieve perfect reproductions of the original content in caches servers, and they allow segmented transfer of content to implement techniques such as *gleaning*. The protocols have been implemented. Their multicast-capability and long-distance functions were experimentally verified. An observation was that long-distance video streaming (e.g. inner-European and trans-Atlantic) is possible and that it is, in terms of file transfer, more efficient and robust but less fair than standard-compliant TCP implementations. While the protocol suite can operate as it is, we have concluded that an augmentation of the long-distance transfers by reservation techniques such as the IntServ of DiffServ approaches would be advantageous; alternatively, a forward-error-correction approach could be added for the communication among servers, at the expense of additional bandwidth.

With the infrastructure in place, the selection of appropriate removal strategies for the cache servers wrapped up the work. The investigation of removal strategies determined a set of rules for a good strategy, and it demonstrates the limits beyond which these strategies need not be tuned. First and foremost, it was established that conditional overwrite (the ability of a cache server to determine that an uncached content is not relevant enough for caching and should rather be passed through to the client) can raise the efficiency of the strategy considerably and allows the support of much larger user populations without increased storage or uplink capacities, compared to unconditional strategies. This ability is rarely found in other application areas of caching since the information that needs to be maintained outgrows the storage gain in most other areas. The typical size of VoD content makes this efficient in our case.

Further investigations demonstrated that Hint based caching can improve this kind of strategy considerably if both the cache size and the user population are small. It is shown that architectures loose performance if cache servers are not dedicated to serving either users or lower level cache servers.

With all of these elements, the possibility to design and implement a wide-area True Video-on-Demand system that is based on decentrally organized caches is demonstrated. In spite of this, the topic is not exhausted by far. Appropriate cooperative strategies can be investigated, better user models are needed and, as a precondition for commercial success, more research activity in the area of copyright protection for untrusted multiparty communication is required.

9. References

[AlAm96] K. C. Almeroth, M. H. Ammar, "On the Use of Multicast Delivery to Provide a Scalable and Interactive Video-on-Demand Service", IEEE Journal on Selected Areas in Communication, 14(6), pp. 1110-1122, August 1996

[AMK97] E. Amir, S. McCanne, R. Katz, "Receiver-driven Bandwidth Adaptation for Light-weight Sessiosn", ACM Multimedia 1997, Seattle, WA, USA, November 1997

[AOG91] D. P. Anderson, Y. Osawa, R. Govindan. Real-Time Disk Storage and Retrieval of Digital Audio/Video Data. TR UCB/CSB 91/646, University of California, Berkely, September 1991.

[AWY96] C. C. Aggarwal, J. L. Wolf, P. S. Yu, "A Permutation-based Pyramid Broadcasting Scheme for Video-on-Demand Systems", Proc. of ICMCS, pp. 118-126, June 1996

[AnMa97] R. Anderson, C. Manifavas: “Chameleon - A New Kind of Stream Cipher”, Proc. of Fourth workshop on Fast Software Encryption, January 1997. Haifa

[AgGo96] I. Agi, L. Gong: "An Empirical Study of Secure MPEG Video Transmission", ISOC Symposium on Network and Distributed System Security, San Diego, CA, 1996

[BAB95] Scott A. Barnett, Gary J. Anido, H.W. Beadle, "Caching Policies in a Distributed Video on-Demand System", Proc. Australian Telecommunication Networks and Applications Conference 1995, Sydney

[BeBi96] C. Bernhardt, E. Biersack, “The Server Array: A Scalable Video Server Architecture”, in “High-Speed Networking for Multimedia Applications”, Kluwer Academic Publishers, 1996, ISBN 0-7923-9681-2

[BGW97] M. Bär, C. Griwodz, L. Wolf, "Long-term Movie Popularity in Video-on-Demand Systems", In Proceedings of the 5th ACM Int’l Multimedia Conference, pages 349-358, November 1997

[BY95] Birk, Yitzhak. Track Pairing, "A Novel Data Layout for VOD Servers with Multi-Zone Reocrding Disks", In Proceedings of the International Conference on Multimedia Computing and Systems (ICMCS) 95, Washington D.C.,May 1995.

[BZB+97] R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin, Resource ReSerVation Protocol (RSVP), Request for Comments:2205, Network Working Group, 1997

[BoSh95] D. Bonch, J. Shaw, "Collision-Secure Fingerprinting for Digital Data", in Proc. of CRYPTO’95, Springer LNCS 963, pp 452-465, 1995

[CAA93] S. Christodoulakis, D. Anestopoulos, S. Argyropoulos, "Data Organization and Storage Hierarchies in a Multimedia Server", CompCon Spring ’93, Digest of Papers, pp. 596-604, February 1993

[CaHu99] Y. Cai, K. A. Hua, "An Efficient Bandwidth-Sharing Technique for True Video on Demand Systems", ACM Multimedia 1999, Orlando, FL, USA, November 1999

[Chen92] Y.S.Chen, “Mathematical modelling of empirical laws in computer application: A case study”, Comput. Math. Applicat., pp. 77-78, Oct. 1992

[Cher94] A. L. Chervenak, "Tertiary Storage: An Evaluation of New Applications", PhD thesis, University of California, Berkely, 1994

[Chu96] Soon M. Chung (Ed.). Multimedia Information Storage and Management. Kulwer Academic Publishers, Norwell, Mass. 1996. ISBN 0-7923-9764-9.

[CKY93] Mon-Song Chen, Dilip D. Kandlur, Philip S. Yu. Optimization of the Grouped Sweeping Scheduling (GSS) with Heterogeneous Multimedia Streams.Proceedings of ACM MM '93, Anaheim, CA, August, 1993, pp. 235-242.

[CaLo97] S. W. Carter, D. Long, "Improving Video-on-Demand Server Efficiency through Stream Tapping", Proc. of ICCCN'97, Las Vegas, NV, USA, September 1997

[CLG+94] Peter M. Chen, Edward K. Lee, Garth A. Gibson, Randy H. Katz, David A. Patterson, "RAID: High-Performance, Reliable Secondary Storage", *ACM Computing Surveys*, 26,2:145-185, June 1994.

[CTC+96] Chen, S. M. Tan, R. H. Campbell, Y. Li, "Real Time Video and Audio in the World Wide Web", in World Wide Web Journal, Volume 1, January 1996.

[CT97] Shenze Chen, Manu Thapar, "A Novel Video Layout Strategy for Near-Video-on-Demand Servers", Proc. of the International Conference on Multimedia Computing and Systems (ICMCS) 97, pages 37-45, Ottawa, June 3-6, 1997.

[DaSi93] A. Dan and D. Sitaram, "Buffer Management Policy for an On-Demand Video Server", RC 19347, IBM Research Division, 1993

[DaSi95] A. Dan, D. Sitaram, "A Generalized Interval Caching Policy for Mixed Interactive and Long Video Workloads", RC 20206 (89404), IBM Research Division, September 1995

[DaSi95b] A. Dan and D. Sitaram, "An Online Video Placement Policy based on Bandwidth to Space Ratio (BSR)", In Proceedings of the 1995 SIGMOD, San Jose, California, May 22-25, pages 376-385, 1995.

[DBS+99] J. Dittmann, A. Behr, M. Stabenau, P. Schmitt, J. Schwenk, J. Ueberberg, "Combining Digital Watermarks and Collision Secure Fingerprints for Digtial Images", in Proc. of the SPIE Conference on Electronic Imaging '99, Security and Watermarking of Multimedia Contents, 24-29 January 1999, SAN Jose, USA, Proc. of SPIE Vol. 3657:51, pp. 171-182, 1999

[Ditt99] J. Dittmann, "Digitale Wasserzeichen", Springer-Verlag, Heidelberg, Germany, 2000, ISBN 3-540-66661-3

[DHH+93] L. Delgrossi, C. Halstrick, D. Hehmann, R. G. Herrtwich, O. Krone, J. Sandvoss, C. Vogt, "Media Scaling for Audiovisual Communications with the Heidelberg Transport System", in Proceedings of ACM Multimedia 93, pp. 99-104, August 1993.

[DKS95] A. Dan, M. Kienzle, D. Sitaram, "A Dynamic Policy of Segment Replication for Load-Balancing in Video-on-Demand Servers", *Multimedia Systems*, 3:93-103, 1995.

[DSST94] A. Dan, P. Shahabuddin, D. Sitaram, D. Towsley, "Channel Allocation under Batching and VCR Control in Video-On-Demand Systems", IBM Research Report, RC 19588, Sept. 1994.

[DSS96] A. Dan, D. Sitaram, P. Shahabuddin: “Dynamic Batching Policies for an On-Demand Video Server”, Multimedia Systems 4(3), pp. 112-121, 1996

[DFV97] L. D’Ardia, B. Fadini, G. Ventre: “Analysis, Classification and Simulation of Multimedia Traffic Sources”, Proceedings of DDCN ‘97, Disitributed Computer Communication Networks, Tel Aviv, November 1997

[EV98] D. Eager, M. Vernon, "Dynamic Skyscraper Broadcasts for Video-on-Demand", Proc. of MIS’98, Istanbul, Turkey, September 1998

[EVZ99] D. Eager, M. Vernon, J. Zahorjan, "Optimal and Efficient Merging Schedules for Video-on-Demand Servers", ACM Multimedia 1999, Orlando, FL, USA, November 1999

[FJL+97] S. Floyd, V. Jacobson, C. Liu, S. McCanne, and L. Zhang, "A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing". IEEE/ACM Transactions on Networking, Volume 5, Number 6, pp. 784-803, December 1997

[GeCh92] J. Gemmell, S. Christodoulakis, "Principles of Delay-Sensitive Multimedia Data Storage and Retrieval", ACM Transactions on Information Systems, 10(1), pp. 51-90, January 1992

[GMDS98] C. Griwodz, O. Merkel, J. Dittmann, R. Steinmetz, "Protecting VoD the Easier Way", In Proc. of ACM Multimedia 1998, pages 21-28, September 1998.

[GLM96] L. Golubchik, J. C. S. Lui, R. R. Muntz, "Adaptive Piggybacking: A Novel Technique for Data Sharing in Video-on-Demand Storage Servers", Multimedia Systems 4, pp. 140-155, 1996.

[GLZS99] C. Griwodz, M. Liepert, M. Zink, and R. Steinmetz, "Tune to Lambda Patching", ACM Performance Evaluation Review, 27(4), pp. 20-26, March 2000.

[GZL+00] C. Griwodz, M. Zink, M. Liepert, G. On and R. Steinmetz, "Multicast Savings in Cache-based Video Distribution", to be presented at MMCN2000, San Jose, CA, USA, January 2000

[GZT99] L. Gao, Z.-L. Zhang, D. Towsley, "Catching and Selective Catching: Efficient Latency Reduction Techniques for Delivering Continuous Multimedia Streams", ACM Multimedia 1999, Orlando, FL, USA, November 1999

[GH94] D. James Gemmell, Jiawei Han, "Multimedia Network File Servers: Multi-Channel Delay Sensitive Data Retrieval", Multimedia Systemes 1(6), pp. 240-252, 1994.

[GhKi96] Shahram Ghandeharizadeh and Dongho Kim. On-line Reorganization of Data in Scalable Continous Media Servers. In Proceedings of Database and Expert Systems Applications 1996, Zurich, Switzerland, pages 751-768, 1996.

[GIZ96] Shahram Ghandeharizadeh, Doug Ierardi, and Roger Zimmermann. An Algorithm for Disk Space Management to Minimize Seeks. Information Processing Letters, 1996.

[GKS95] Shahram Ghandeharizadeh, Seon Ho Kim, and Cyrus Shahabi. Continuous Display of Video Objects Using Multi-Zone Disks. TR 94-592, USC, April 1995.

[HaSc95] Haskin, R.: The Shark Continuous-Media File Server, Proc. of IEEE COMPCON, Feb. 1993.

[HCS98] K. A. Hua, Y. Chai, S. Sheu, "Patching: A Multicast Technique for True Video-on-Demand Services", Proc. of ACM Multimedia 1998, pp. 191-200, Bristol, England, September 1998

[HS97] K. A. Hua, S. Sheu, "Skyscraper Broadcasting: A New Broadcasting Scheme for Metropolitan Video-on-Demand Systems", Proc. of ACM SIGCOMM'97, pp. 89-100, September 1997

[Hes98] C. K. Hess, "Media Streaming Protocol: An Adaptive Protocol for the Delivery of Audio and Video over the Internet", Master Thesis, University of Illinois at Urbana-Champaign, 1998, http://choices.cs.uiuc.edu/Papers.html

[Huf52] D. A. Huffman: "A Method for the Construction of Minimum Redundancy Codes", In Proceedings of IRE 40, S. 1098-1101, September 1952

[Hil87] W. Hilberg: Digitale Speicher 1, Oldenburg Verlag, ISBN 3-486-20270-7 (1987)

[ISO93] ISO/IEC IS 11172, "Information Technology - Coding of Moving Pictures and Associated Audio for Digital Storage Media up to about 1.5 MBit/s", ISO/IEC JTC1/SC29, 1993

[ISO93b] ISO/IEC IS 10918, "Information Technology - Digital Compression and Coding of Continuous-Tone Still Images", ISO/IEC JTC1/SC29, 1993

[ISO96] ISO/IEC IS 13818, "Information Technology - Generic Coding of Moving Pictures and Associated Audio Information (MPEG2)", ISO/IEC JTC1/SC29, 1996

[ISO98] ISO/IEC CD 14496-6, "Information Technology - Generic Coding of Moving Pictures and Associated Audio Information (MPEG4)", ISO/IEC JTC1/SC 29 WG 11 N2206, 1998

[ITU96] ITU-T, "Recommendation H.263: Video Coding for Low Bitrate Communication", 1996

[JT97] L. Juhn, L. Tsend, "Harmonic Broadcasting for Video-on-Demand Service", IEEE Transactions on Broadcasting, 43(3), pp. 268-271, September 1997

[JG97] J. Jarmasz, N. D. Georganas: "Designing a Distributed Multimedia Synchronization Scheduler", Proc.IEEE Multimedia Systems'97, Ottawa, June 1997

[KSSW00] M. Karsten, J. Schmitt, B. Stiller, L. Wolf, "Charging for packet-switched network communication - motivation and overview", Computer Communications, 23(3); 290-302, Jan. 2000

[KLC97] J. Kim, Y. Lho, K. Chung. An Effective Video Block Placement Scheme on VOD Server based on Multi-Zone Reocrding Disks. In Proceedings of the International Conference on Multimedia Computing and Systems (ICMCS) 97, pages 29-36, Ottawa, June 3-6, 1997.

[Kor97] Jan Korst. Random Duplicated Assignment: An Alternative to Striping in Video Servers. In Proceedings of the 5th ACM Int'l Multimedia Conference, pages 219-226, Seattle, Nov. 9-13, 1997.

[Kra88] S. Krakowiak. Principles of Operating Systems. MIT Press, Cambridge, 1988.

[KVL97] Rajesh Krishnan, Dinesh Venkatesh, Thomas D. C. Little. A Failure and Overload Tolerance Mechanism for Continuous Media Servers. Proceedings of the ACM MM 97 Conference, pp. 131-142, 1997.

[KRSB97] T. Kunkelmann, R. Reinema, R. Steinmetz, T. Blecher: Evaluation of Different Video Encryption Methods for a Secure Multimedia Conferencing Gateway, Proc. of the 4th Int'l COST237 Workshop, Lisboa, Portugal, Dezember 1997, pp. 75-89

[KVMW98] T. Kunkelmann, H. Vogler, M. L. Moschgath, L. Wolf, "Scalable Security Mechanisms in Transport Systems for Enhanced Multimedia Services", Proc. of 3rd European Conference on Multimedia Apllications, Services and Techniques (ECMAST'98), Berlin, Germany, 1998

[Kun98] T. Kunkelmann, "Sicherheit für Videodaten", Disserationsschrift (PhD thesis), Vieweg-Verlag, ISBN 3-528-05680-0, 1998

[KüGr98] C. Küfner, C. Griwodz, "Möglichkeiten für den Einsatz von Lastverteilungsstrategien verteilter Systeme in der Videoverteilung", Technical Report 07, KOM, TU-Darmstadt, November 1998

[LeKa91] E. K. Lee, R. H. Katz. Performacen consequences of parity placement in disk arrays. In Proceedings of the 4th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-IV). IEEE, New York, 190-199. 1991.

[LLG98] Peter W. K. Lie, John C. S. Lui, Leana Golubchik. Threshold-Based Dynamic Replication in Large-Scale Video-on-Demand Systems. Accepted for RIDE 98.

[LoSh93] P. Lougher, D. Shepherd. The Design of a Storage Service for Continuous Media. The Computer Journal, 36(1), pp. 32-42, 1993.

[Litt61] J. D. C. Little, "A Proof of the Queueing Formula L=lW", Operations Research, No.9 pp. 383-387, 1961

[LiVe94a] T. D. C. Little, D. Venkatesh, "Prospects for Interactive Video-on-Demand", IEEE Multimedia, Vol. 1, No. 3, Fall 1994, pp. 14-24

[LiVe94b] T. D. C. Little, D. Venkatesh, "Popularity-based Assignment of Movies to Storage Devices in a Video-on-Demand System", ACM Multimedia Systems, 1994

[Lia98] T, Liao, "Light-weight Reliable Multicast Protocol", Technical Report, INRIA, Le Chesnay Cedex, France, 1998

[Lamp98] Bernd Lamparter - personal communication

[LiPa96] J.C. Lin, S. Paul, "RMTP: A Reliable Multicast Transport Protocol", INFOCOM 1996

[MaRo98] I. Marshall, C. Roadknight, "Linking Cache Performance to User Behaviour", 3rd Int'l WWW Caching Workshop, Manchester, England, June 15-17, 1998

[MJ95] S. McCanne, V. Jacobson, vic: A Flexible Framework Framework for Packet Video. ACM Multimedia'95, November 1995, San Francisco, CA, pp. 511-522.

[MNO+94] C. Martin, P. S. Narayan, B. Özden, R. Rastogi and A. Siberschatz, The Fellini Multimedia Storage Server, in Chung: Multimedia Information Storage and Management, Kluwer Academic Publishers, 1994

[MNO+96] C. Martin, P. S. Narayanan, B. Ozden, R. Rastogi, A. Silberschatz, "The Fellini Multimedia Storage Server", in [Chu96], pp. 117-146, 1996.

[MS95] T. B. Maples, G. A. Spanos: “Performance Study of a Selective Encryption Scheme for the Security of Networked, Real-Time Video”, Proc. of the 4th Int’l Conf. on Computer Communications and Networks, Las Vegas, Nevada, September 1995

[NPSS95] J.P. Nussbaumer, B.V. Patel, F. Schaffa, J.P.G. Sterbenz, "Networking Requirements for Interactive Video on Demand", IEEE Journal on Selected Areas in Communication, 13(5), pp.779-787, 1995

[NBT97] J. Nonnenmacher, E. Biersack, D. Towsley. Parity-Based Loss Recovery for Reliable Multicast Transmission. ACM SIGCOMM 1997, Cannes, France, September 1997

[OuDo89] John Ousterhout, Fred Douglis. Beating the I/O Bottleneck: A case for Log-Structured File Systems. Operating Systems Review, 23(1), pp. 11-28, 1989.

[PCL98a] J.-F. Paris, P. E. Mantey, D. Long, "Efficient Broadcasting Protocols for Video on Demand", Proc. of MASCOTS’98, pp. 127-132, July 1998

[PCL98b] J.-F. Paris, P. E. Mantey, D. Long, "A Low Bandwidth Broadcasting Protocol for Video on Demand", Proc. of ICCCN’98, pp. 690-697, October 1998

[PLM99] J.-F. Paris, D. Long, P. E. Mantey, "Zero-Delay Broadcasting Protocols for Video-on-Demand", ACM Multimedia 1999, Orlando, FL, USA, November 1999

[PeWe72] E. W. Peterson, E. J. Weldon. Error-Correcting Codes. 2nd ed. MIT Press, Cambridge, Mass, 1972.

[PGK88] David. A. Patterson, Garth Gibson, Randy H. Katz. A Case for Redundant Arrays of Inexpensive Disks (RAID). Proceedings of the 1988 ACM Conference on Management of Data (SIGMOD), Chicago, IL, pp. 109-116, June 1988.

[Pos81] Jon Postel, Transmission Ccontrol Protocol (TCP), USC/Information Sciences Institute, Request for Comments: 793, 1981

[Pos80] Jon Postel, User Datagram Protocol (UDP), Request for Comments: 768, 1980

[QNT97] L. Qiao, K. Nahrstedt, I. Tam: “Is MPEG Encryption Using Random Lists instead of Zig Zag Order Secure?“, IEEE International Symposium on Consumer Electronics, December 1997, Singapore

[RaRa91] R. Ramarao, V. Ramamoorthy, "Architectural Design of On-Demand Video Delivery Systems: The Spatio-Temporal Storage Allocation Problem", IEEE ICC, 1991

[ReWy93] A. L. Narasimha Reddy, Jim Wyllie. Disk scheduling in a multimedia I/O system. Proceedings of ACM MM ‘93, Anaheim, CA, August, 1993, pp. 225-223.

[ReWy94] A. L. Narasimha Reddy, Jim Wyllie. I/O Issues in a Multimedia System. COMPUTER, 27(3), pp. 69-74, 1994.

[RFC1633] R. Braden, D. Clark, S. Shenker, "Integrated Services in the Internet Architecture: an Overview", IETF, RFC 1633, June 1994

[RFC1819] L. Delgrossi, L. Berger (Edtrs.), "Internet Stream Protocol Version 2 (ST2) Protocol Specification - Version ST2+", IETF, RFC 1819, August 1995.

[RFC1889] H. Schulzrinne, S. Casner, R. Frederick, "RTP: A Transport Protocol for Real-Time Applications. IETF Audio/Video Transport Working Group", IETF, RFC 1889, January 1996

[RFC2205] R. Braden, Ed., L. Zhang, S. Berson, S. Herzog, S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, IETF, September 1997

[RFC2326] H. Schulzrinne, A. Rao, R. Lanphier, "Real Time Streaming Protocol (RTSP)", RFC 2326, IETF, April 1998

[RFC2327] M. Handley, V. Jacobson, "SDP: Session Description Protocol", RFC 2327, IETF, April 1998

[RFC2391] P. Srisuresh, D. Gan, "Load Sharing using IP Network Address Translation (LSNAT)", RFC2391, IETF, August 1998

[RFC2608] E. Guttman, C. Perkins, J. Veizades, M. Day, "Service Location Protocol, Version 2", RFC 2608, IETF, June 1999

[RFC2474] K. Nichols, S. Blake, F. Baker, D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, IETF, December 1998

[RFC2475] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, "An Architecture for Differentiated Services", RFC 2475, IETF, December 1998

[RoBi98] P. Rodriguez, E.W.Biersack, "Bringing the Web to the Network Edge: Large Caches and Satellite Distribution", In WOSBIS'98. ACM/IEEE MobiCom Workshop on Satellite-based Information Services, Dallas, USA. October 30, 1998.

[SGRV97] Prashant J. Shenoy, Pawan Goyal, Sriram S. Rao, Harrick M. Vin. Symphony: An Integrated Multimedia File System. In Proceedings of SPIE/ACM Conference on Multimedia Computing and Networking (MMCN'98), San Jose, CA, pp. 124-138, Jan 1998.

[SAW97] C. Shahabi, M. H. Alshayeji, S. Wang, "A Redundant Hierarchical Structure for a Distributed Continuous Media Server, in Proc. of the IDMS'97, September 1997.

[SDW92] W.T. Strayer, B.J. Dempsey, A.C. Weaver, "XTP: The Xpress Transfer Protocol", Addison-Wesley, Reading, Mass 1992, ISBN 0-201-56351-7, more recent documents still available at http://www.ca.sandia.gov/xtp/biblio.html

[SoLe97] P. Soderquest, M. Leeser, "Optimizing the Data Cache Performance of a Software MPEG-2 Video Decoder", ACM Multimedia 1997, Seattle, WA, USA, November 1997

[Sch96] B. Schneier, "Angewandte Kryptographie - Protokolle, Algorithmen und Sourcecode in C", Addison-Wesley, Bonn, ISDN 3-8931-9854-7, 1996

[SKK+90] M. Satyanarayanan, J. Kistler, P. Kumar, M.E. Okasaki, E.H. Siegel, D. C. Steere. Coda: A highly available file system for a distributed workstation environment. IEEE Transactions on Computers. 39:441-459, 1990.

[Sta95] W. Stallings, "Sicherheit in Datennetzen", Prentice-Hall, München, ISBN 3-930436-29-9, 1995

[SBD96] B. Sabata, M.J. Brown, and B.A. Denny, "Transport Protocol for Reliable Multicast: TRM", IASTED International Conference on Networks, 1996

[TeFl95] W. H. Tetzlaff and R. Flynn, "Elements of Scalable Video Servers", Proc. of COMPCON 1995, pp. 239-248, 1995.

[TKS94] W. Tetzlaff, M. Kienzle, D. Sitaram, “A Methodology for Evaluating Storage Systems in Distributed and Hierarchical Video Servers”, COMPCON 94 Conference Proceedings, pp.430-439

[TMD95] Renu Tewari, Rajat Mukherjee, Daniel M. Dias, "Real-Issues for Clustered Multimedia Servers", IBM Research Division, RC 20108, 6/20/95

[Tewa98] R. Tewari, "Architecture and Algorithms for Scalable Wide-area Information Systems. Dissertation", University of Texas, Austin, TX, USA, August 1998

[Tan96] L. Tang: “Methods for Encrypting and Decrypting MPEG Video Data Efficiently”, Proc. of the 4th ACM Multimedia Conference, Boston, MA, November 1996, pp. 219-230

[VI96] S. Viswanathan, T. Imielinski, "Metropolitan Area Video-on-Demand Service using Pyramid Broadcasting", Multimedia Systems, 4(4), pp. 197-208, 1996

[VeLi95] D. Venkatesh, T. D. C. Little. Dynamic Service Aggregation for Efficient Use of Resources in Interactive Video Delivery. Proceedings of the 5th NOSSDAV conference, pp. 113-116, Nov. 1995.

[WaDu97] Yuewei Wang, David H. C. Du. Weighted Striping in Multimedia Servers. In Proceedings of the International Conference on Multimedia Computing and Systems (ICMCS) 97, pages 102-109, Ottawa, June 3-6, 1997.

[YCK92] P. S. Yu, M.-S. Chen, D. D. Kandlur. Design and Analysis of a Grouped Sweeping Scheme for Multimedia Storage Management. 3rd Int’l Workshop on Network and Operating System Support for Digitial Audio and Video (NOSSDAV92), San Diego, Nov. 1992.

[ZiGh97] Roger Zimmermann and Shahram Ghandeharizadeh. Continuous Display Using Heterogeneous Disk-Subsystems. In Proceedings or ACM Multimedia 97, November 8-14, Seattle, pages 0-0, 1997.

HTTP References

[h:MeGa94] J. Meyer, F. Gadegast, "Sicherheitsmechanismen fü Multimediadaten am Beispiel MPEG-1 Video", Technical Report, TU Berlin, http://www.mpeg1.de/doc/secmpeg.ps.gz, 1994

[h:CNCL96] Communication Networks, Aachen University of Technology, "Communication Networks Class Library", in Annual Report March 1994 - July 1996,
http://www.comnets.rwth-aachen.de/

[h:MaCa98] Marimba, "Castanet Infrastructure Suite Datasheet", Marimba Product Resource Library, 1998,
http://www.marimba.com/products/resource.htm

[h:WCW99] The 4th International Web Caching Workshop, San Diego, California, March 31 - April 2, 1999; URL: wwwcache.ja.net/events/workshop

[h:Dvb98] DVB Project, 1998
http://www.dvb.org

[h:Davi98] Digital Audio-Visual Council, 1998
http://www.davic.org

[h:Real99] Real Networks, 1999
http://www.real.com

[h:Hos98] Philipp Hoschka (Edtr.), "Synchronized Multimedia Integration Language (SMIL) 1.0 Specification", W3C Recommendation 15-June-1998, W3C, June 1998
http://www.w3.org/TR/1998/REC-smil-19980615

[h:SaDe94] J. Sandvoss, L. Delgrossi (Edtrs.), "The Berkom II MultiMedia Transport System (MMT) Version 4.0",
http://www.prz.tu-berlin.de/docs/html/prot/mmt1/mmt1.html

[h:Stre99] StreamingServer.org, "The Portable RT[S]P Internet Streaming Server", after July 1999
http://www.streamingserver.org/priss.html

10. Appendices

10.1 Design of the Overwrite-Capable Multimedia File System

One of our working issues is the design and subsequent implementation of an overwrite-capable multimedia filesystem (tagged OCFS). Since overwriting of data is not a typical feature of multimedia filesystems, the relevance of such an operation is not necessarily clear.

For a filesystem implementation to interact with a reliable multicast protocol, it is necessary that lost data is recovered and the transferred file is finally available completely. With protocols such as TCP or some reliable multicast protocols, a windowing mechanim is used that retransmits the lost packets quickly in such a way that gaps can be filled in a main memory buffer of the receiving system. The drawback of such windowing protocols is the unreliable end-to-end delay on the application level that renders them unuseful for real-time stream transmission unless they are augmented by strict QoS guarantees.

Our reliable variation of RTP is fully compatible with continuous media streaming, but the amount of data that must be expected to arrive at the client before retransmitted data fills the gaps in the original stream of the first transmission precludes the buffering of the arriving data until all gaps are filled. By adding the overwrite feature to our filesystem, we are able to save the arriving data to disk linearly, to leave gaps (zeros) in the file, and to fill in the gaps later.

10.2 Protection from Collusion Attacks

10.2.1 Collusion Protection of the Chameleon Key, 2-party Identification

N = #all bits, k_1 = #marked bits in key B1,k_2 = #marked bits in key B2,$n = k_1 = k_2$

$$P(\text{at least one bit error occurs in both keys}) = 1 - \left.\frac{\binom{k_1}{r} \cdot \binom{N-k_1}{k_2-r}}{\binom{N}{k_2}}\right|_{r=0} = 1 - \frac{\binom{N-n}{n}}{\binom{N}{n}}$$

$$P(\text{at least one bit error occurs in both keys})$$

$$= 1 - \frac{\binom{2^{22}-n}{n}}{\binom{2^{22}}{n}} = 1 - \frac{(2^{22}-n)! \cdot (2^{22}-n)!}{(2^{22}-n-n)! \cdot (2^{22})!} = 1 - \frac{\prod_{i=2^{22}-2n+1}^{2^{22}-n} i}{\prod_{i=2^{22}-n+1}^{2^{22}} i}$$

10.2.2 Collusion Protection of the Chameleon key, 3-party Collusion Attack

:

$$
\begin{aligned}
&P(\text{vote does not remove error}) \\
&= P(\text{key2 hits error in key1 and key3 hits no error in key1}) \\
&+ P(\text{key2 hits no error in key1 and key3 hits error in key1}) \\
&+ P(\text{key2 hits error in key1 and key3 hits error in key1}) \\
&= P(\text{key2 hits error in key1}) \\
&+ P(\text{key3 hits error in key1}) \\
&- P(\text{key2 hits error in key1 and key3 hits error in key1}) \\
&= 2 \cdot P(\text{key2 hits error in key1}) \\
&- P(\text{key2 hits error in key1 and key3 hits error in key1})
\end{aligned}
$$

$P(\text{at least one bit error occurs in two or three keys})$

$$
= 2 \cdot \left(1 - \frac{\binom{2^{22}-n}{n}}{\binom{2^{22}}{n}} \right) - \left(1 - \frac{\binom{2^{22}-n}{n}}{\binom{2^{22}}{n}} \cdot \frac{\binom{2^{22}-n}{2n}}{\binom{2^{22}}{2n}} \right)
$$

$$
= 2 \cdot \left(1 - \frac{(2^{22}-n)! \cdot (2^{22}-n)!}{(2^{22}-2n)! \cdot (2^{22})!} \right) - \left(1 - \frac{(2^{22}-n)! \cdot (2^{22}-n)!}{(2^{22}-n-n)! \cdot (2^{22})!} \cdot \frac{(2^{22}-2n)! \cdot (2^{22}-n)!}{(2^{22}-3n)! \cdot (2^{22})!} \right)
$$

$$
= 2 \cdot \left(1 - \frac{\prod_{i=2^{22}-2n+1}^{2^{22}-n} i}{\prod_{i=2^{22}-n+1}^{2^{22}} i} \right) - \left(1 - \frac{\prod_{i=2^{22}-2n+1}^{2^{22}-n} i}{\prod_{i=2^{22}-n+1}^{2^{22}} i} \cdot \frac{\prod_{i=2^{22}-3n+1}^{2^{22}-2n} i}{\prod_{i=2^{22}-n+1}^{2^{22}} i} \right)
$$

10.3 Cost Calculations for Distribution in Binary Trees

The calculations for the costs of the different distribution models that are compared in “Motivation of Gleaning for Caching Hierarchies” on page 39 are based on a binary tree as shown in Figure 17. This binary tree is definitly not an absolutely correct representation of the real world and I do not at all believe that future VoD distribution architectures will look like a binary tree. Video cache servers will certainly not have one up and two downlinks and be located in a router. Rather, I assume that that Internet-style architectures grow to become architectures for VoD.

Nevertheless we deciced to do the cost calculations based on a binary tree because this allows us to model many other network layouts by the fact that some of the links are set to 0. It is obvious that it is not possible to model all possible layouts of VoD architectures. Our model e.g. has the limit that a cache server of a higher level can only be connected to a maximum of two cache Servers in the level below. On the other hand we do this calculation in order to get an upper limit of the costs and are able to show that techniques like caching, Patching, and the combination of both will lower the costs in general. It is our goal to include this techniques in our simulation for video caching in order to receive more detailed resulsts.

We think of a binary tree distribution architecture of depth *d*.

We denote a link in the tree by its level *t* and its index at this level, *n*: E_n^t. If we select an arbitrary link, it is called E^t. Similarly, cache servers are labelled N_n^t and N^t, respectively. For convenience, we consider N^d a client rather than a cache server. We assume that the cost for handling one video stream at each link E_n^t at a level *t* is equal to another at the same level and that the cost for storing a video in each cache server N_n^t at level t is equal to another at the same level. We name these costs for each link or server C_t^E and C_t^N, respectively. For computing the average cost at one level *t*, this value can simply be multiplied with 2^t.

We assume a set of movies $\{m_i | i \varepsilon \{1, \ldots, M\}\}$. All of these movies have the same length, measured in time, L_1 and the same data rate. To express the requirements of cache filling, we assume also an average time L_r after which the popularity of all movies changes and optimal positioning needs to be recalculated.

We assume that each end-user in the system is watching exactly one video at any time.

10.4 Analytical Distribution Model - Binary Tree

For simplification, our example calculations assume binary distribution trees as shown in Figure 17. With appropriate weight and cost settings we can model a limited class of balanced, hierarchical distribution topologies compliant with these assumptions. We are currently working on a more complex and realistic simulation for video caching integrating these techniques in order to receive more detailed results.

We think of a binary tree distribution architecture of depth *d*. We denote a link in the tree by its level *t* and its index *n* at this level: E_n^t. If we select an arbitrary link, it is called E^t. Similarly, cache servers are labeled N_n^t and N^t, respectively. For convenience, we consider N^d a client rather than a cache server. We assume the cost per concurrent video stream to be the same C_t^E for each link E_n^t on one level *t*. Also, we assume the hard disk cost for one video to be C_t^N at each cache server N_n^t on one level *t*. The numbers of links and caches at one level *t*are 2^t.

We assume a set of movies *M*. All of these movies $m \in M$ have the same length, measured in time, L_1 and the same data rate, but possibly different draw probabilities *P(m)*. In caching scenarios, we assume that each cached movie m_i is stored in all caches of one optimally chosen level *t(m)*.

The necessity to have sufficiently large central servers that are able to handle the number of streams that are concurrently requested imposes a cost S_0 for the basic installation of each server, and a cost S_1 for each concurrent stream that is supported by a server. Each end-user in the system is watching exactly one video at any time, ie. $\sum_{m \in M} P(m) = 1$. The number of clients is very big compared to the number of different movies and active (cache) servers, the popularity of movies is constant for all clients. This gives us draw probablities being independent of time and hierarchy location, but also gives the problematic postulation of a majority of inactive, thus zeroed cache servers. We enforce this by defining the base server setup cost S_0 sufficiently high.

10.4.1 Unicast: No Patching, No Caching

The simplest approach to deliver video is the distribution from a central server via unicast. This allows all kinds of video-on-demand features, but is intense in terms of network as well as server load. We calculate costs for such an approach first.

Since there are no movies stored in the caches there will be no storage costs: $C_t^N = 0, \forall t$

Network costs for each currently running movie: $\sum_{t=1}^{d} C_t^E$, which are the cost of a complete link from the central server to the end-user. As every client is watching exactly one movie at any point of time, the overall network cost for streaming is $\sum_{m \in M} \left(P(m) 2^d \sum_{t=1}^{d} C_t^E \right) = 2^d \sum_{t=1}^{d} C_t^E$ The interarrival time is irrelevant in this case, because no streams are shared. With this and a number of clients of 2^d, the central server approach has an overall cost of $S_0 + 2^d \cdot S_1 + 2^d \sum_{t=1}^{d} C_t^E$

10.4.2 Unicast: No Patching, Caching

Network Cost

This implies that networking costs are generated only for the delivery of the movie from the cache server to the clients that are located downstream from this cache server or, in terms of the binary tree, in each subtree with a root node at level *t(i)*. The networking costs for this movie and for this subtree of depth *d-t(m)* can be calculated as in section Unicast: No Patching, No Caching:

$$P(m) \cdot 2^{d-t(m)} \sum_{t=t(m)+1}^{d} C_t^E$$

Although the formula concerning the distribution probability of the movies does still apply in this case (the sum of probabilities equals 1), this should not be integrated into this formula, because the optimal level *t(m)* is different for each movie, depending on its probability.

Server Cost

Since there are $2^{t(m)}$ cache servers at level *t(m)*, the above networking cost occurs $2^{t(m)}$ times. The cost generated by the movie *m* that is stored at level *t(m)* is then

$$2^{t(m)} \cdot P(m) \cdot 2^{d-t(m)} \sum_{t=t(m)+1}^{d} C_t^E = P(m) \cdot 2^d \sum_{t=t(m)+1}^{d} C_t^E$$

The resulting storage cost for a movie *m* on all cache servers at level *t(m)* is $2^{t(m)} C_{t(m)}^N$

The cost of the capacity needed by this cache server depends on the average number of concurrent streams it has to serve for each movie *m*. This is calculated from the hit probability of the movie and the number of clients that the cache server serves. The setup cost for a needed cache server on level *t* is $S_0 + S_1 \cdot 2^{d-t} \cdot \sum_{m \in M} P(m) \cdot \delta(t(m) = t)$ where $\delta(p) = \begin{cases} 1, & p \text{ is true} \\ 0, & p \text{ is false} \end{cases}$

A cache server has to be set up if its level is the optimal cache level for any movie, thus installation cost for serving clients is as . $\sum_{t=0}^{d-1} \left[2^t \delta\left(\bigcup_{m \in M} (t(m) = t) \right) \cdot \left(S_0 + S_1 2^{d-t} \sum_{m \in M} P(m) \cdot \delta(t(m) = t) \right) \right]$

As we assume a constant system state, there are no cost to store or stream movies on the root server cached elsewhere.

Simplified and increased by the network cost, this gives the following formula for the overall cost for our model with caching:

$$\left[\sum_{t=0}^{d-1} \left(2^t \cdot \bigcup_{m \in M} \delta_{t(m)}(t) \right) \right] \cdot S_0 + 2^d \cdot S_1 + \sum_{m \in M} \left[P(m) 2^d \sum_{t=t(m)+1}^{d} C_t^E + 2^{t(m)} C_{t(m)}^N \right]$$

10.4.3 Greedy Patching with central server

The simplest form of Patching is Greedy Patching without buffering limits at the clients. Besides the fact that clients will be overly expensive when they are built to buffer complete movies, we have shown in [GLZS99] that the optimal restart time in terms of server load depends on *P(m)* and thus, the largest required buffer does not need to hold a complete movie.

However, we assume this kind of Patching to find an approximation for the cost of a distribution system. Assume a binary distribution tree of depth *d*, caching is not applied in this tree.

For each movie *m*, we define $\eta_m = P(m)$ for ease of reuse of the formulas.

Server effort

Since this approach is using a central server, S_0 is needed only once. The number of streams that need to be served concurrently is also reduced in comparison to the unicast case with a central server. The formula is derived as in Section , and yields the setup cost, the basic server cost for multicast streams of *m* and the total cost of unicast patch streams: $S_0 + (1-(1-\eta_m)^{2^d}) \cdot S_1 + 2^{d-1} S_1$

Multicast portion

First, we try to calculate the network load that is generated at each level of the binary tree due to the probability of a joint stream for multiple clients; ie. we want to find a formula for savings of network bandwidth in the upper levels of the binary tree. We assume a random distribution of the clients that share a stream of movie *m* in the overall set of clients. The probability of a network link to be involved in a multicast playout of a specific movie is the probability, that any client below demands that specific movie. This probability is

$$P(E_n^t \text{ not involved}) = 1 - P(E_n^t \text{ serves } m)$$

$$= 1 - \prod_{i=1}^{2^{d-t}} P(N_{k_i}^d \text{ does not request } m) = 1 - \prod_{i=1}^{2^{d-t}} P(N^d \text{ does not request } m) = 1 - (1-\eta_m)^{2^{d-t}}$$

which means that at each level *t*, an average of $(1-(1-\eta_m)^{2^{d-t}}) \cdot 2^t$ links are involved in the same multicast of movie *m*, and a cost that is generated at level t by the multicast streams is $(1-(1-\eta_m)^{2^{d-t}}) \cdot 2^t \cdot C_t^E$

Unicast portion

At the same time, the unicast patches need to be distributed to the clients. These unicast patches require a direct transmission from the central server to the end-user, and this unicast transmission behaves mainly like a regular video transmission according to Section 10.4.1. The major difference is that the length of a unicast patch is less than a full length video transmission rather the length of the unicast patch is on average 1/2 of the patching window, which is in this case the full movie length ([GLZS99]). Thus, the load of unicast streams at level *t* is in this case: $\frac{1}{2}(\eta_m 2^d C_t^E) = \eta_m \cdot 2^{d-1} \cdot C_t^E$

Overall cost

When the unicast and multicast formulas are combined, the overall cost at level *t* is $[2^t \cdot (1-(1-\eta_m)^{2^{d-t}}) + 2^{d-1} \cdot \eta_m] \cdot C_t^E$

and the overall cost of distribution of all movies, through the whole tree is the summation

$$S_0 + \left[2^{d-1} + \sum_{m \in M} (1-(1-\eta_m)^{2^d})\right] \cdot S_1 + \sum_{m \in M} \sum_{t=1}^{d} \left(2^t - 2^t \cdot (1-\eta_m)^{2^{d-t}} + 2^{d-1} \cdot \eta_m\right) \cdot C_t^E$$

Savings Compared To Unicast With Central Server

The average Greedy Patching case is not costlier than the unicast with central server. With the inequality

$$\forall t \le d, \forall m \in M \text{ with } P(m) < 1:$$
$$1-(1-P(m))^{2^{d-t}} \le 1-(1-P(m)) = P(m) \le P(m)2^{d-t-1}$$

the comparison of the server efforts to Section 10.4.1 gives a possible saving: $\left[2^{d-1} + \sum_{m \in M} (1-(1-\eta_m)^{2^d})\right] \cdot S_0 \le 2^d \cdot S_1$

Together with the comparison of network load below this is a first hint to integrate Patching in the delivery system.

$$\sum_{m \in M} \sum_{t=1}^{d} \left[2^t - 2^t \cdot (1-P(m))^{2^{d-t}} + 2^{d-1} \cdot P(m)\right] \cdot C_t^E \le \sum_{m \in M} \sum_{t=1}^{d} C_t^E P(m) 2^d = 2^d \sum_{t=1}^{d} C_t^E$$

10.4.4 Patching with limited buffer and central server

When the restart rate *r(m)* for the multicast stream of a specific movie *m* is increased, i.e., the window size to covered by patch streams is reduced, then the probability that clients receive the same multicast is reduced, but the use of a limited patching window size realistically limits the needed buffer size at the client. As in [GLZS99], we assume for simplicity that the multicast transmissions are repeated regularly, and that the length of such a cycle is called the restart time. The restart time here is expressed as a portion of the movie length: $\frac{1}{r(m)} L_1$ The probability for a client to join a specific multicast playout of a specific movie *m* follows as $\eta_m = \frac{1}{r(m)} \cdot P(m)$

Server effort

With a patchiong window of $\frac{1}{r(m)} L_1$, we calculate the average number of concurrent unicast patches to be served according Section . Ie. that the number of concurrent unicast streams for *m* is $\frac{1}{2} \cdot 2^d \cdot \frac{P(m)}{r(m)} = 2^{d-1} \eta_m$

This yields the number of concurrent unicast streams that need to be supported by the central server at each time. Unlike for Greedy Patching, *r(m)* is assumed to be optimal but different for different *m*. The server cost for unicast streams is $2^{d-1} S_1 \sum_{m \in M} \eta_m$ which is inverse proportional to the restart rate! The server cost per movie *m* for multicast streams is increasing with the restart rate: $r(m)(1-(1-\eta_m)^{2^d}) \cdot S_1$

Multicast portion

The multicast cost of the distribution system is calculated as in the Section , with the redefined η_m. As *r(m)* copies of the stream can be active at any time, the average load at level *t* is $r(m) \cdot (1-(1-\eta_m)^{2^{d-t}}) \cdot 2^t \cdot C_t^E$

Unicast portion

The computation of the unicast load of the distribution system is the same as in the last section, but with the reduced average length of the unicast patch streams, the values differ. With the redefined value η_m, however, the formula remains the same as in the previous section, the cost a level *t* is $\eta_m \cdot 2^{d-1} \cdot C_t^E$

Overall cost

The combined costs of elements yield the average cost for a distribution system that uses Patching with a central server and movie-dependent window sizes for the delivery of unicast patch streams.

$$S_0 + \left[\sum_{m \in M} (2^{d-1} \cdot \eta_m + r(m)(1-(1-\eta_m)^{2^d})) \right] \cdot S_1$$
$$+ \sum_{m \in M} \left[\sum_{t=1}^{d} (C_t^E (\eta_m \cdot 2^{d-1} + r(m) \cdot (1-(1-\eta_m)^{2^{d-t}}) \cdot 2^t)) \right]$$

10.4.5 Patching with Caching

We assume that for large hierarchies, savings can be increased by combining patching with caching. To verify this, we start with the inner part of the formula from Section . We assume that for each movie *m* there is exactly one level *t(m)*, where this movie is cached in all servers. We calculate the server cost for one cache server for *m*. The depth of the distribution sub-tree is *d-t(m)*. Analogous to Section 10.4.4, for this movie, the effort to support streams on the cache server (without basic setup S_0 and the movie storage cost C_t^N, which can be calculated as in Section) and on the network links below this server is given by

$$S_1(2^{d-t(m)-1} + r(m)(1-(1-\eta_m)^{2^{d-t(m)}})) + \sum_{k=1}^{d-t(m)} (C_k^E (\eta_m \cdot 2^{d-t(m)-1} + r(m) \cdot (1-(1-\eta_m)^{2^{d-t(m)-k}}) \cdot 2^k))$$

This cost occurs once for each server at this level, and that cost, in turn, needs to be calculated once for each movie *m*. This results in an overall cost for Patching with caching of

$$\sum_{m \in M} \left[2^{t(m)} \cdot \left(S_1(2^{d-t(m)-1} + r(m)(1-(1-\eta_m)^{2^{d-t(m)}})) \right.\right.$$
$$\left.\left. + \sum_{k=1}^{d-t(m)} (C_k^E (\eta_m \cdot 2^{d-t(m)-1} + r(m) \cdot (1-(1-\eta_m)^{2^{d-t(m)-k}}) \cdot 2^k)) \right) \right]$$
$$+ S_0 \cdot \sum_{t=0}^{d-1} \left(2^t \cdot \delta(\bigcup_{m \in M} (t = t(m))) \right) + \sum_{m \in M} (2^{t(m)} C_{t(m)}^N)$$

10.5 Simulation Details

10.5.1 The Hit Rate Effect of Daytime Variations: Increasing numbers of movies[1]

The input script for the simulation series is given in Figure 58. The content type specifies that “Baer” style movies will be produced that die after $MOVIES days. The popularity of these titles undergoes some variation from the Baer style to introduce daytime variations. 50% of the movie popularity depends fully on the Baer parameters and the other half is modified by a sinus curve. The phase of of the sinus curve has its peak either close to 14:00 or close to 20:00 simtime.

```
PARAMETERS
{
     SEED    $RAND

    CONTENT TYPE MOVIE
      { BAER * { 0.5 + { SINUS( . 14.0 ) | SINUS( . 20.0 ) } } }
     * DEATH($MOVIES)

     CONTENT PREPARE $MOVIES

     LOG START $MOVIES + 3
     LOG GROUP 1 CONN "atms"
     LOG GROUP 2 SERVER "he"
     TERMINATE $MOVIES + 15
}
```

Figure 58: Parameter setting for 10.5.1

In the nodes definition, you may notice the user type “ManyUser”. Since users in the applied scenarios are always connected through ADSL and receive at most one stream, their link is never overloaded. To reduce main memory during simulation, these user have been

1. task7.4, IncMC155 and IncMC622

combined into the new type “ManyUser”. Given enough uplink bandwidth, the behaviour of this user type is identical to the model of single users.

```
NODES
{
    NODE World {
        TYPE World
    }

    NODE User {
        TYPE ManyUser
        SET COPIES $USERS
    }

    NODE HeadEnd {
        TYPE Server
        SET CAPACITY $CAPACITY GB
        SET STRATEGY IRG1 ( "eternal", "conditional", "time" )
        SET STATGROUP 2
    }
}
```

Figure 59: Node descriptions for 10.5.1

```
CONNECTIONS
{
    CONNECTION ATM {
        TYPE Bus;
        SET THROUGHPUT 622 MBIT/S;
        SET LENGTH 30 MS;
        SET STATGROUP 1
    }

    CONNECTION ADSL {
        TYPE P2P
        SET THROUGHPUT 150000 MBIT/S
        SET LENGTH 5 MS
    }
}
```

Figure 60: Network descriptions for 10.5.1

```
HIERARCHY
{
      World -> ( mirdochegal ) {
          ATM -> 1 * {
            HeadEnd -> ( mirnochimmeregal ) {
              ADSL -> User
            }
          }
      }
}
```

Figure 61: Topology for 10.5.1

With this test, the influence of an increasing number of concurrently available Baer movies on the hit ratio and throughput was examined. The figures 62-65 demonstrates the results. The general impression that an increasing number of concurrently available movie titles will result in a reduced efficiency of a cache is trivial. The observation that doubling of the user population has hardly any effect on the quality of the cache filling is less trivial.

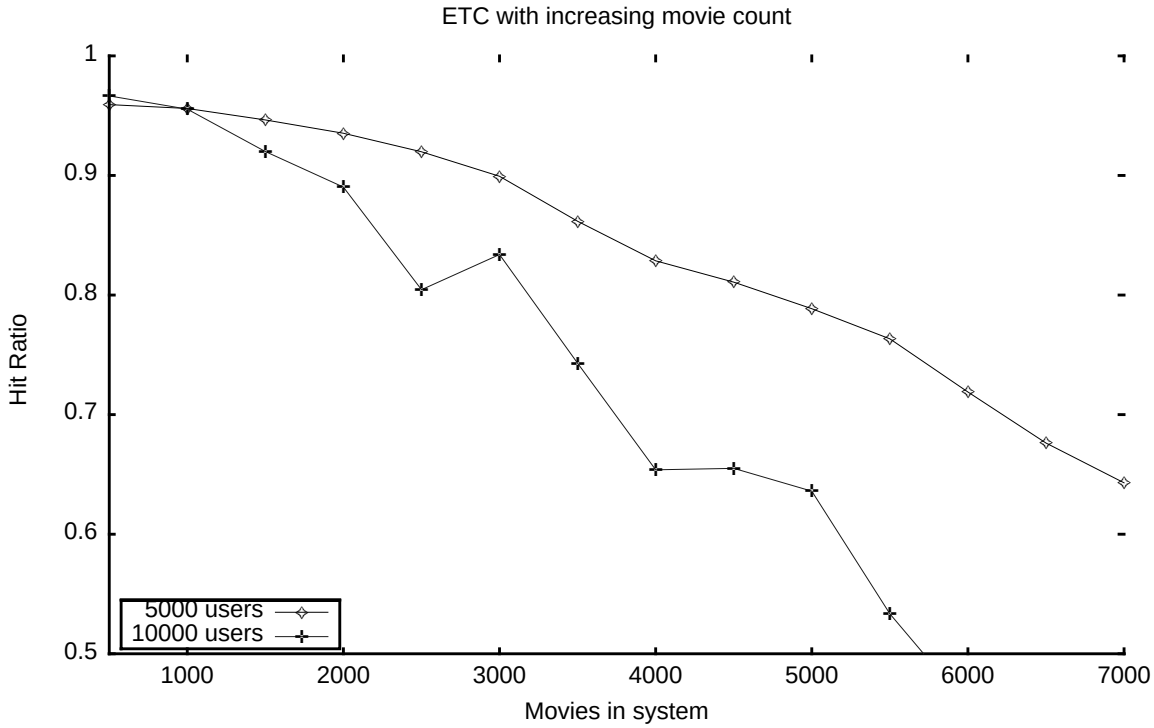


Figure 62: Decrease in HitRatio with increasing number of available movies, uplink capacity 155 MBit/s

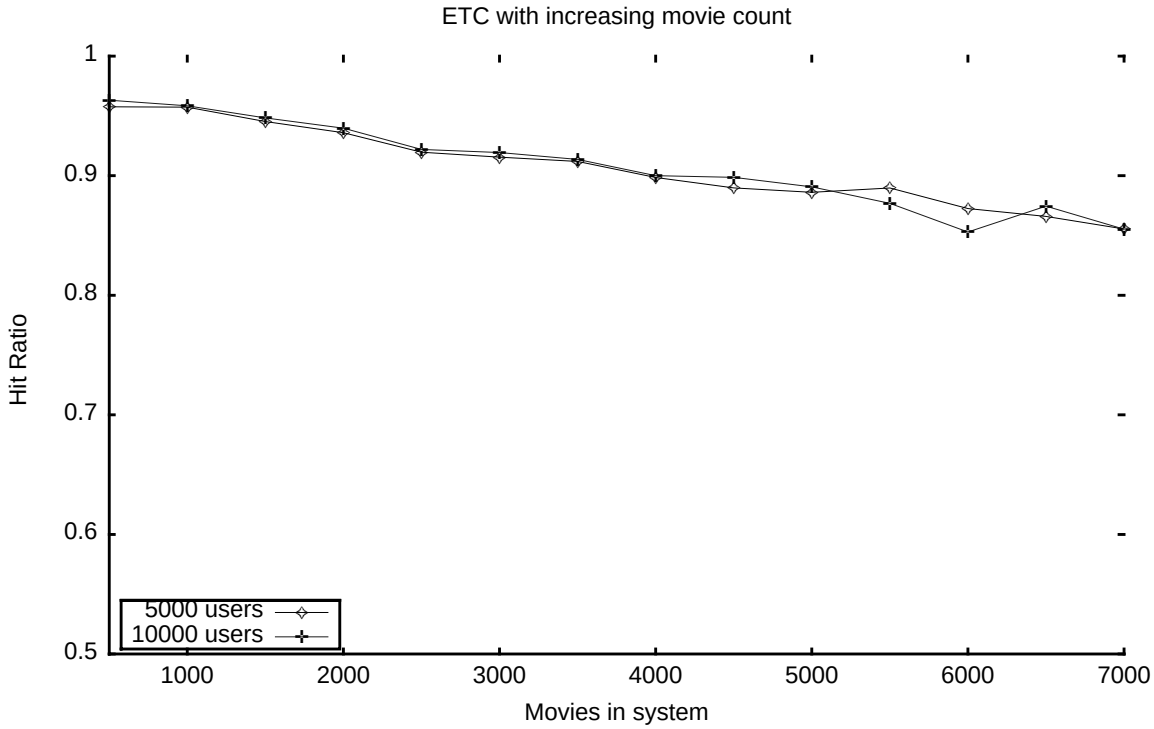


Figure 63: Uplink requirements increase with number of available movies, uplink capacity 155 MBit/s

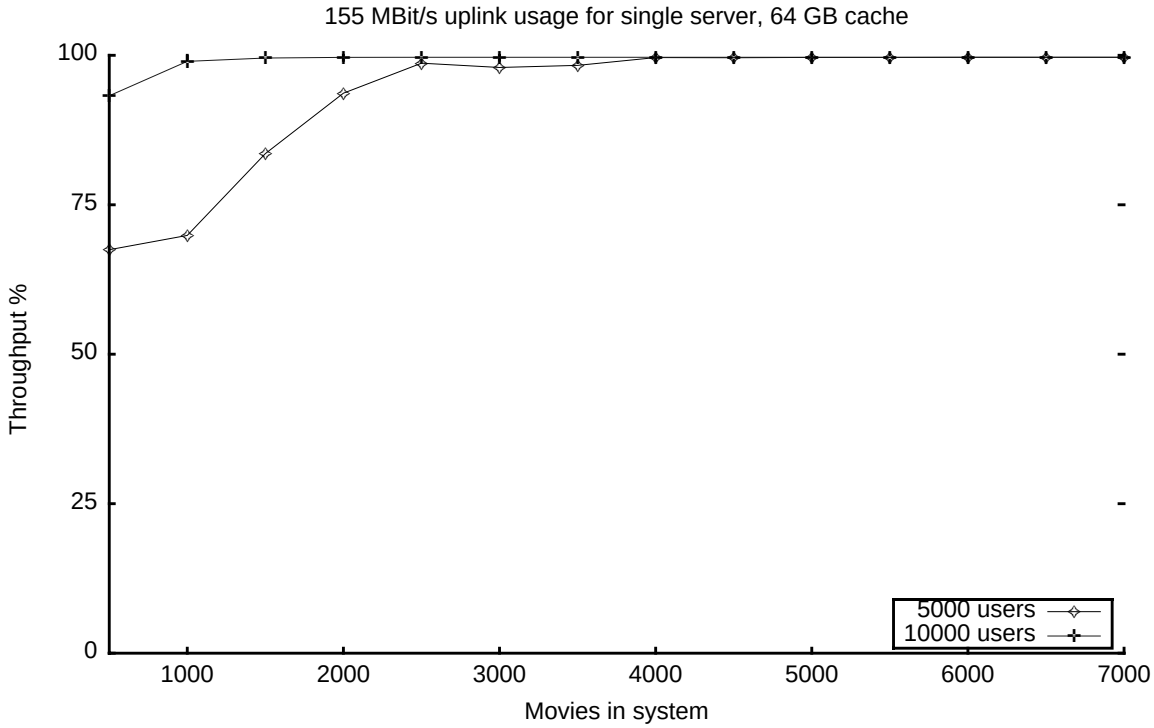


Figure 64: Uplink requirements increase with number of available movies, uplink capacity 155 MBit/s

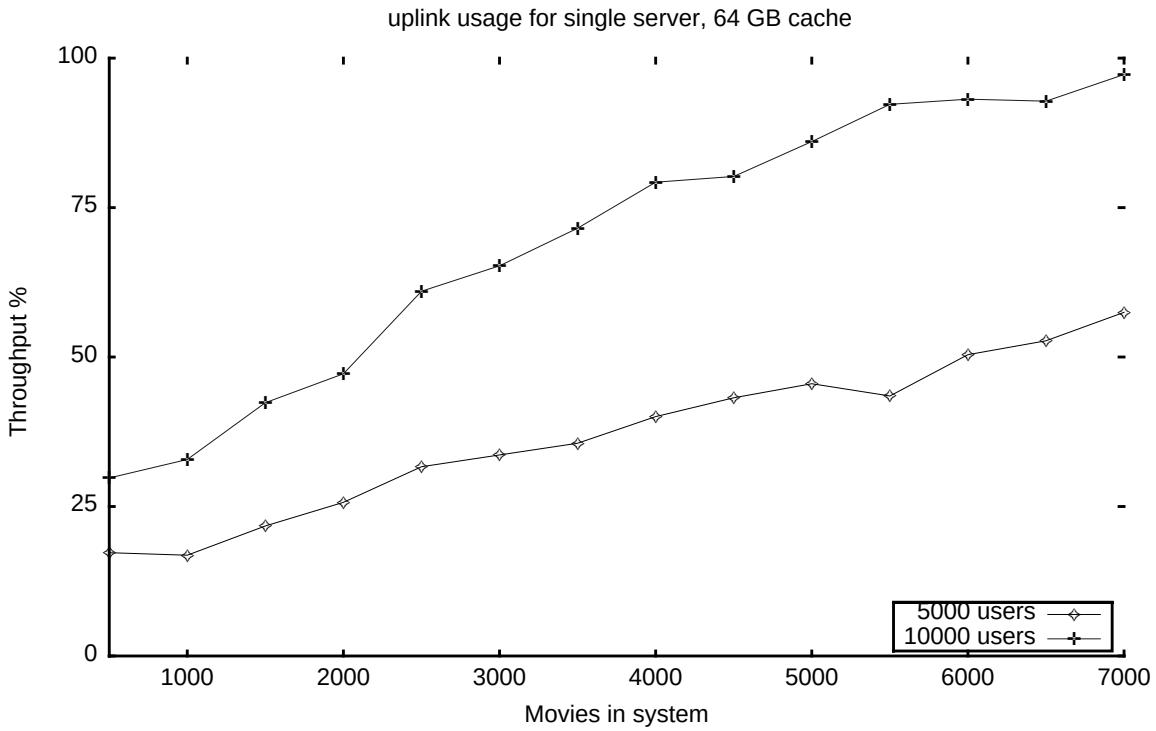


Figure 65: Uplink requirements increase with number of available movies, uplink capacity 622 MBit/s

10.5.2 The Hit Rate Effect of Daytime Variations: Random sinus wave

The daytime variation modeling is obscure due to the lack of a proven model of real-world user behaviour. Several potential sources have been considered and discarded due to the lack or realism. Broadcast television uses a self-fulfilling model by assigning times of the day to certain genres. Rental stores do not collect data on the actual viewing times of their customers. Hotel television has a very restricted programme. Pay-TV channels like Premiere World in Germany would be an appropriate source of information, but right now this information is not released.

To examine the basic properties of daytime variations, we decided to apply additional variations with a 24 hours cyclic behaviour to the long-term popularity curves of our movie model. We start with a sinus function that is applied to 50% of the long-term popularity value at each time. Figure 66 shows the parameter section of the simulator configuration file. Within a 24 hours period, the popularity of a title varies between 50% and 150% of the popularity that is defined by the long-term model. For Section 10.5.2, the daily popularity peak of each titles is chosen independently throughout the day.

To save computing time, the popularities are not recomputed for each selection that is made by a user, but only every 10 minutes (simulated minutes).

The results can be seen in Figures 67 through 74. By themselves, these figures are no more but another demonstration of the superiority of conditional a overwrite strategy.

```
PARAMETERS
{
      SEED    $RAND
     CONTENT TYPE MOVIE { BAER * { 0.5 + SINUS } } * DEATH(500)
      CONTENT PREPARE 500

      LOG START 503
      LOG GROUP 1 CONN "atms"
      LOG GROUP 2 SERVER "he"
      TERMINATE 515
}
```

Figure 66: Parameter setting for 10.5.2

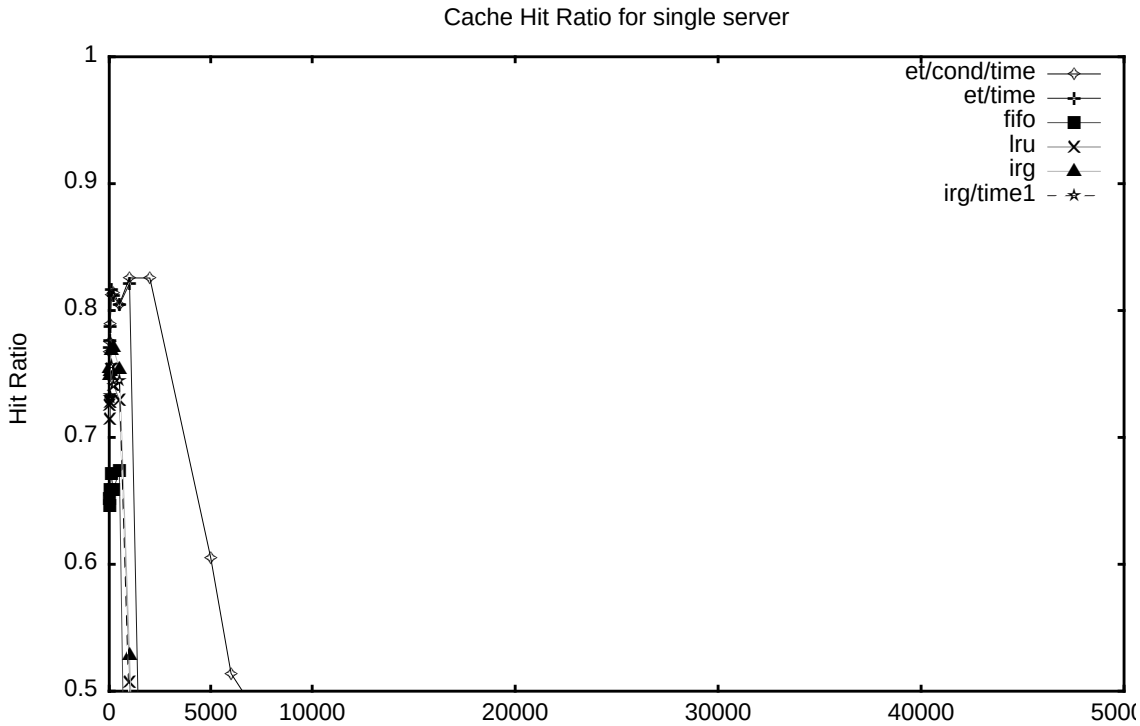


Figure 67: Hit ratio development with increasing number of users, 155 MBit/s uplink, 32 GB cache

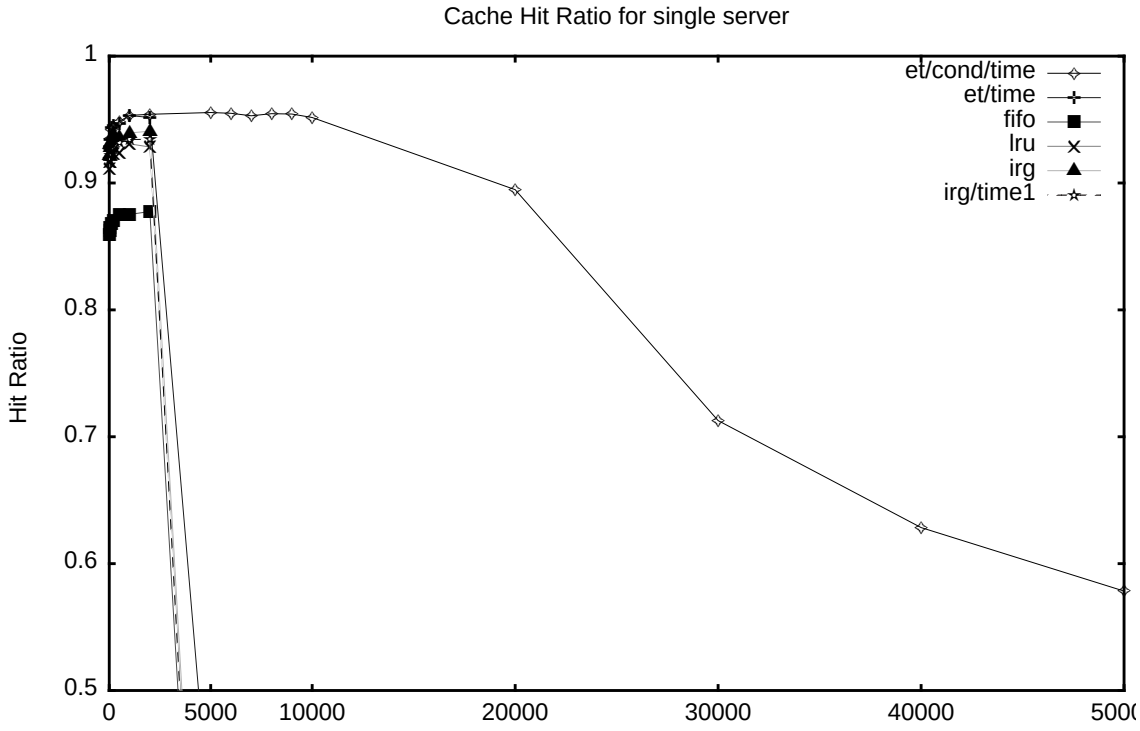


Figure 68: Hit ratio development with increasing number of users, 155 MBit/s uplink, 64 GB cache

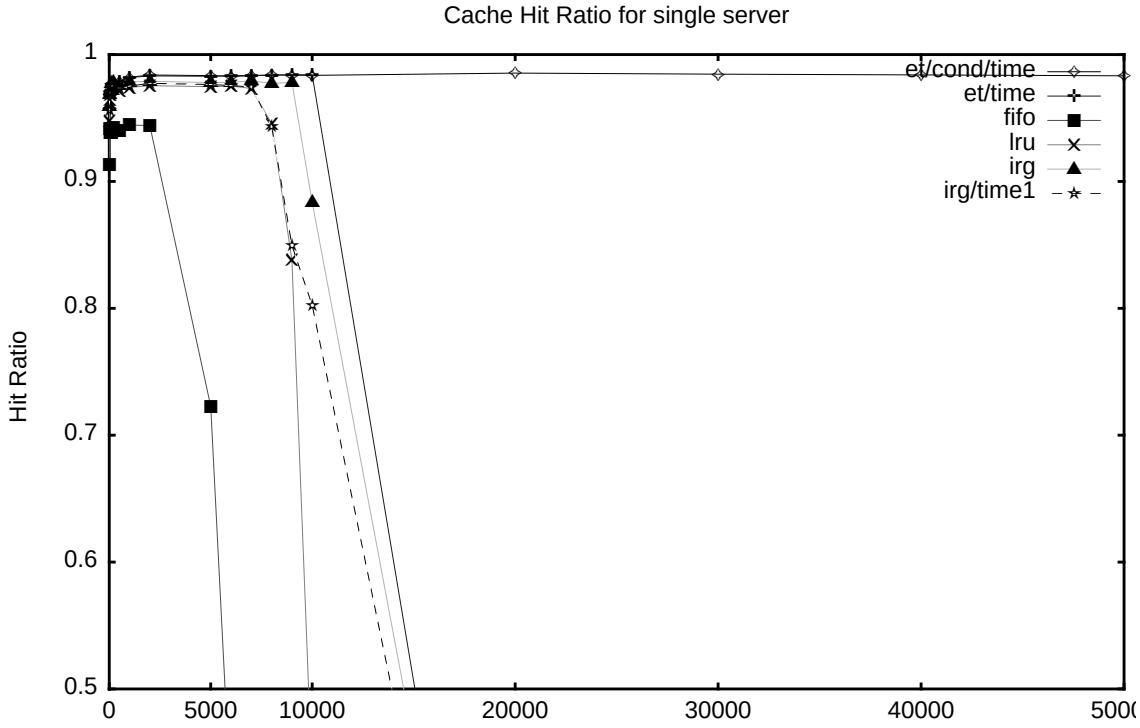


Figure 69: Hit ratio development with increasing number of users, 155 MBit/s uplink, 96 GB cache

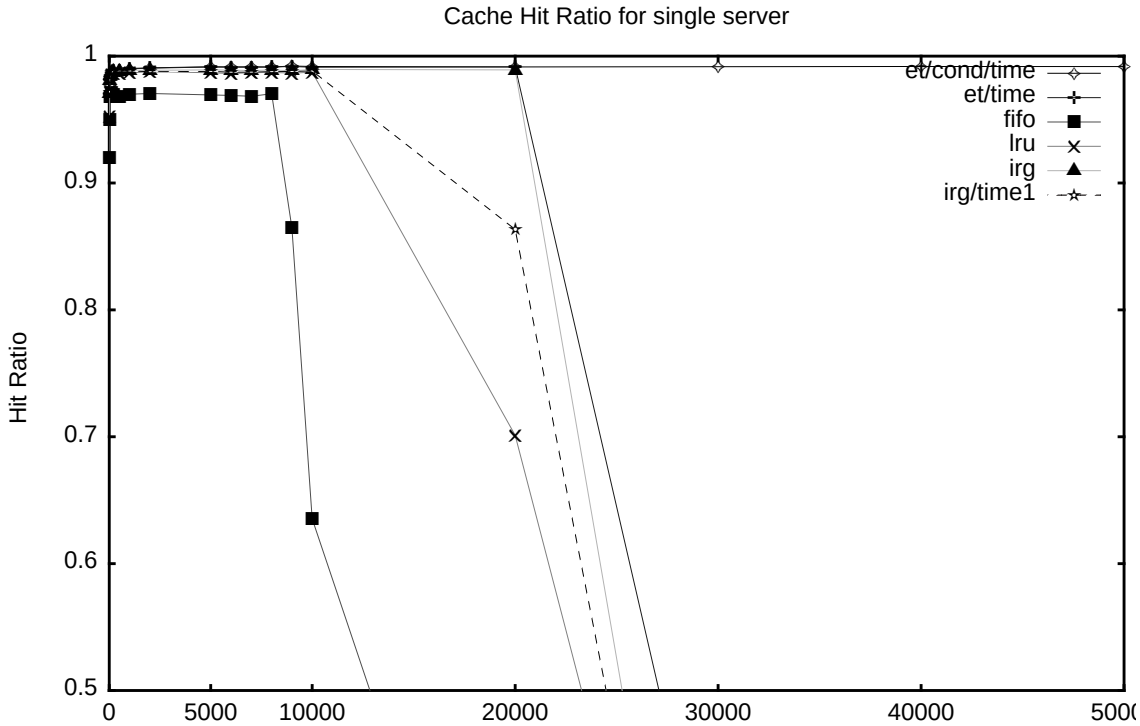


Figure 70: Hit ratio development with increasing number of users, 155 MBit/s uplink, 128 GB cache

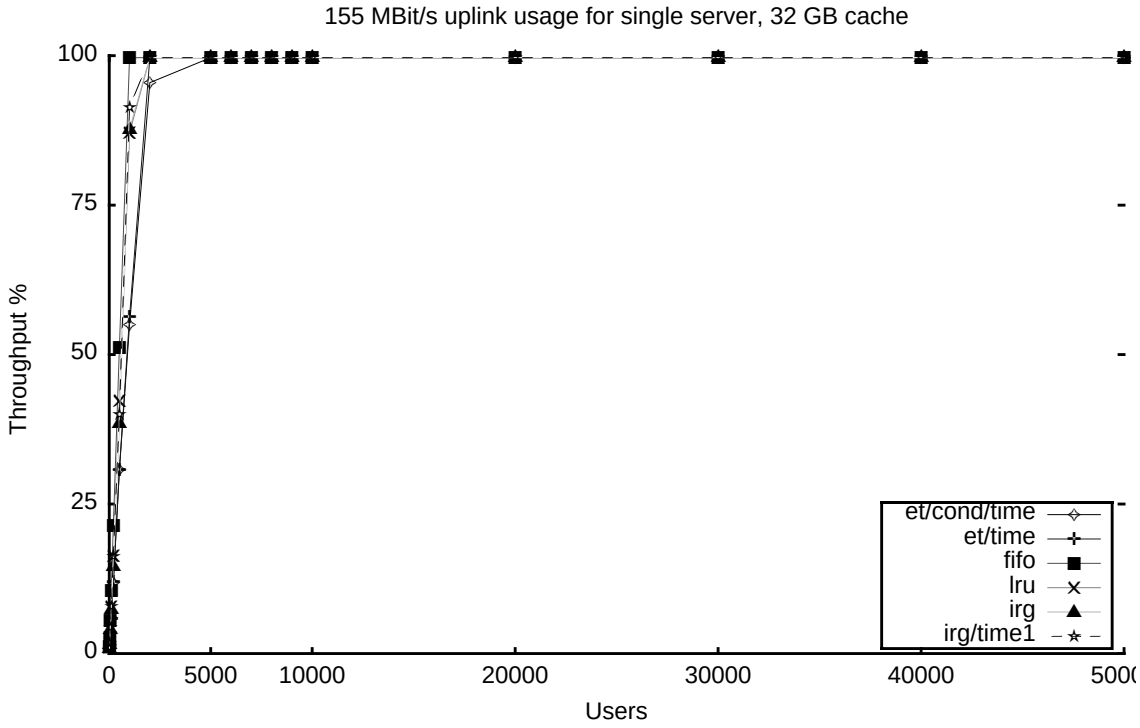


Figure 71: Throughput development with increasing number of users, 155 MBit/s uplink, 32 GB cache

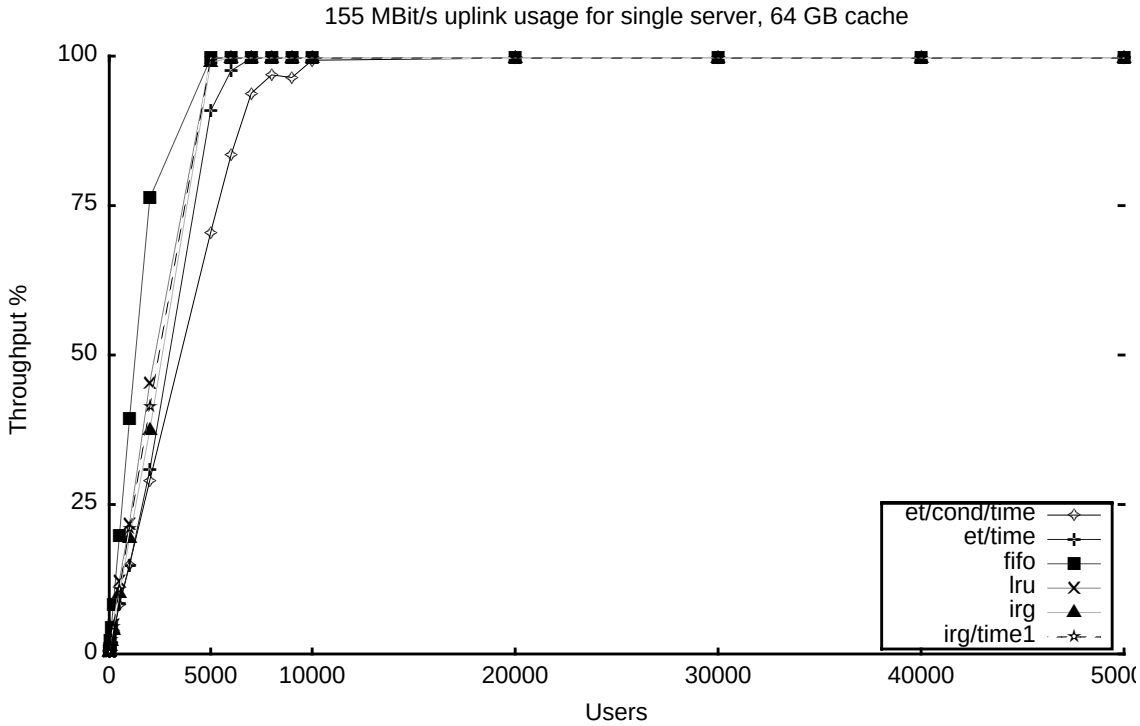


Figure 72: Throughput development with increasing number of users, 155 MBit/s uplink, 64 GB cache

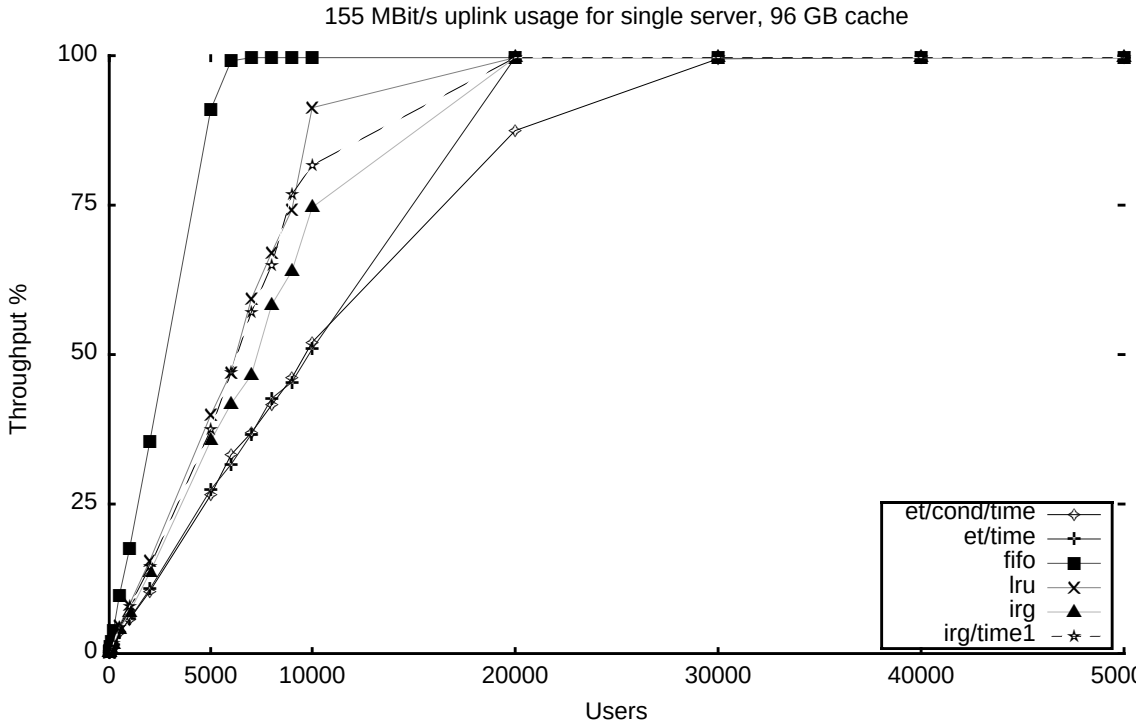


Figure 73: Throughput development with increasing number of users, 155 MBit/s uplink, 96 GB cache

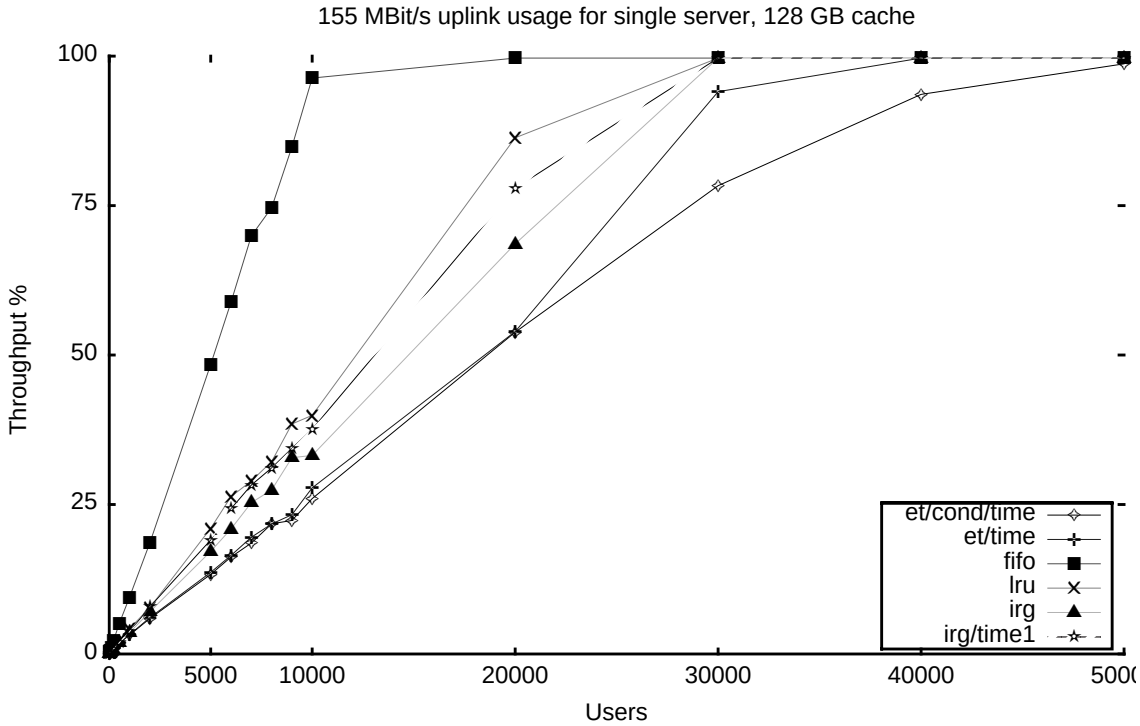


Figure 74: Throughput development with increasing number of users, 155 MBit/s uplink, 128 GB cache

10.5.3 The Hit Rate Effect of Daytime Variations: Two hot spots

It is probably more realistic than total freedom of choice that users prefer certain genres at certains days of time. This would result in concentrated requests on movies in a cyclic manner. Some titles are reaching peek popularity in the morning, and some in the evening. To account for this, we have added simulations that create two daily hot spots. A movie would have its daily peek popularity either close the one of the other hot spot. The following simulation locates these hot spots at 14:00 and 20:00 hours. Figure 75 shows the simulation parameters.

```
PARAMETERS
{
      SEED    $RAND
      CONTENT TYPE MOVIE
        { BAER * { 0.5 + { SINUS( . 14.0 ) | SINUS( . 20.0 ) } } } * DEATH(500)
      CONTENT PREPARE 500

      LOG START 503
      LOG GROUP 1 CONN "atms"
      LOG GROUP 2 SERVER "he"
      TERMINATE 515
}
```

Figure 75: Parameter setting for 10.5.3

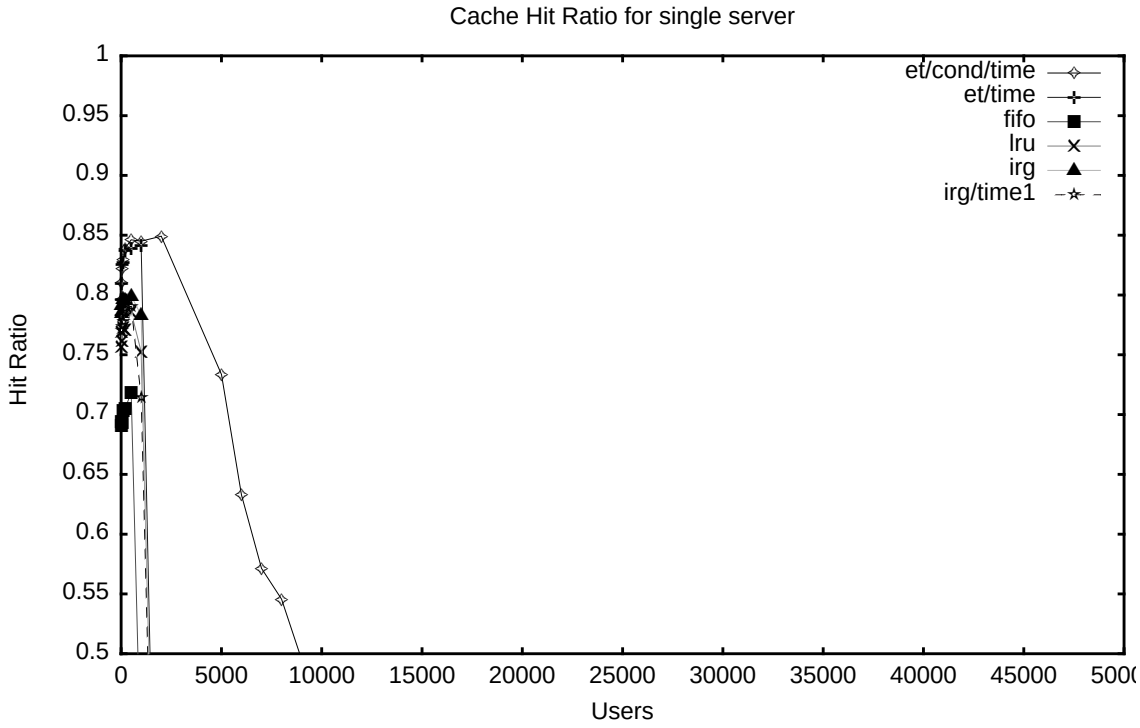


Figure 76: Hit ratio development with increasing number of users, 155 MBit/s uplink, 32 GB cache

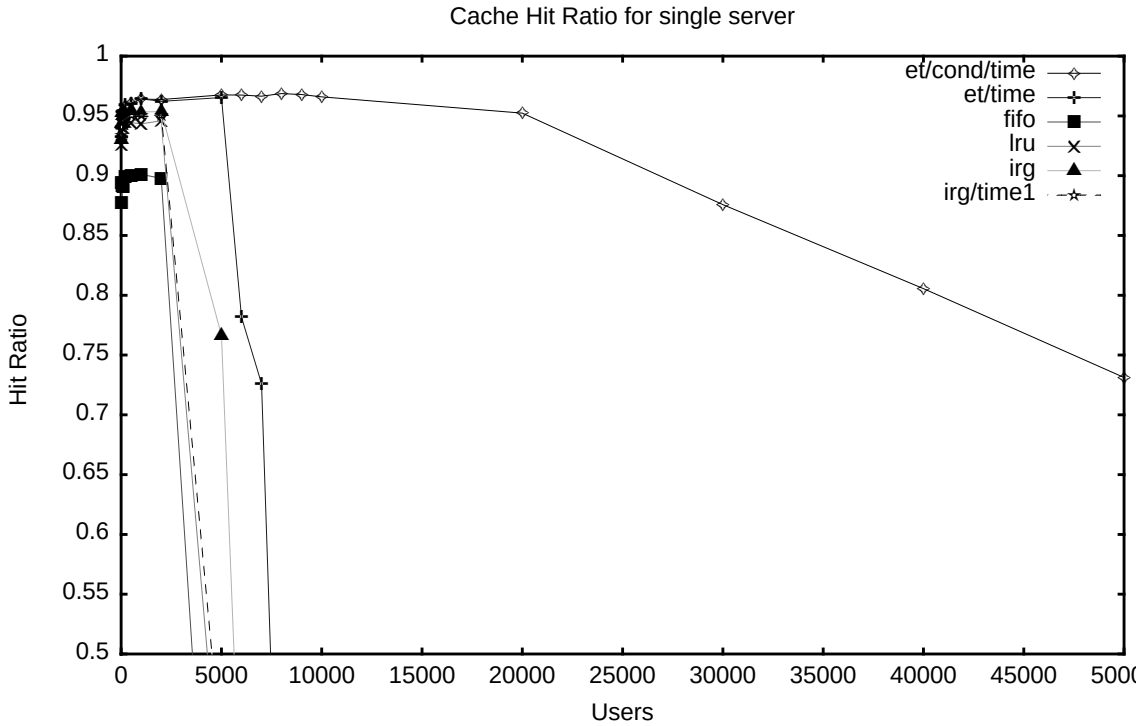


Figure 77: Hit ratio development with increasing number of users, 155 MBit/s uplink, 64 GB cache

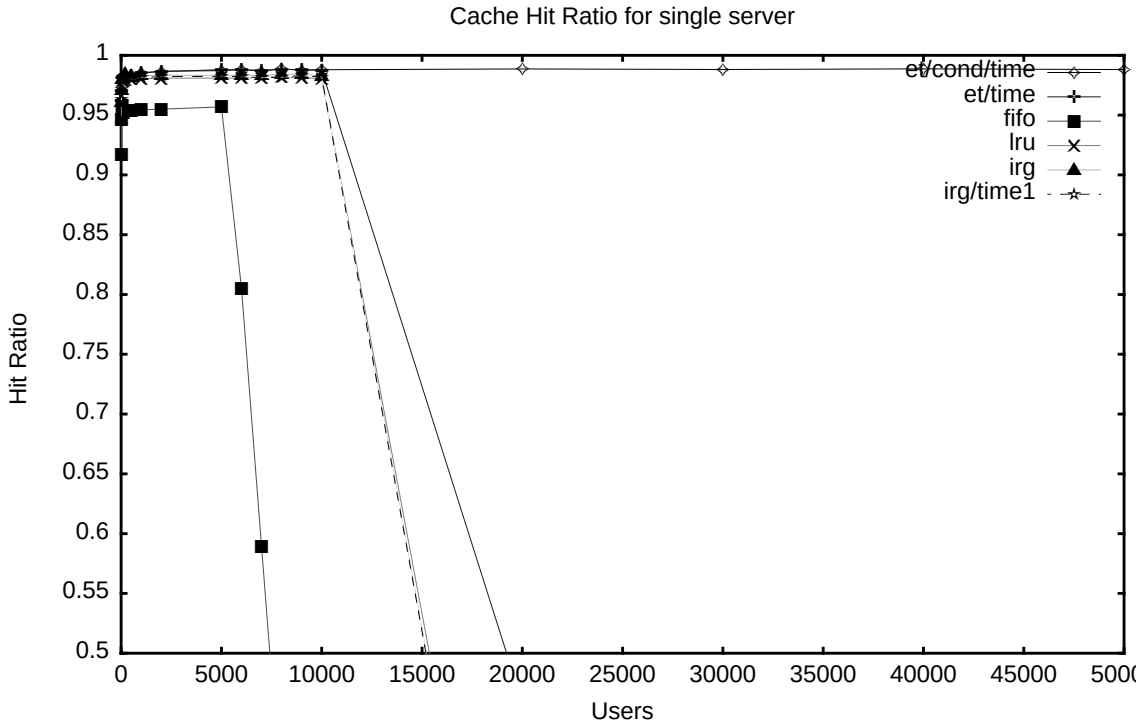


Figure 78: Hit ratio development with increasing number of users, 155 MBit/s uplink, 96 GB cache

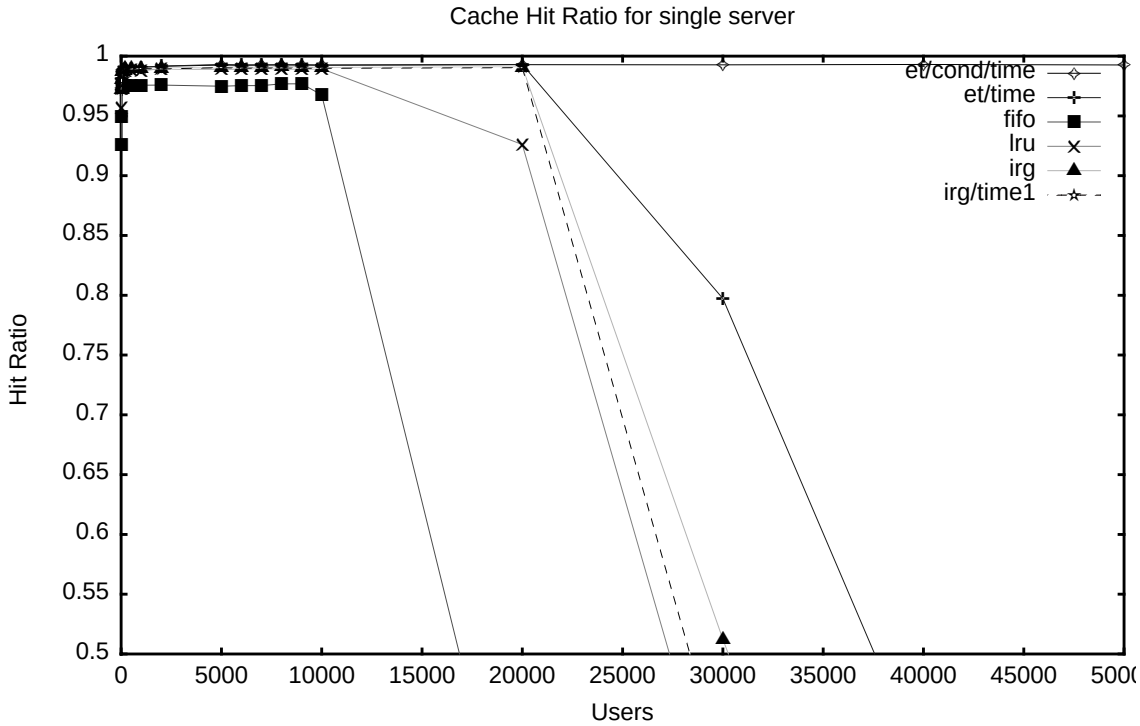


Figure 79: Hit ratio development with increasing number of users, 155 MBit/s uplink, 128 GB cache

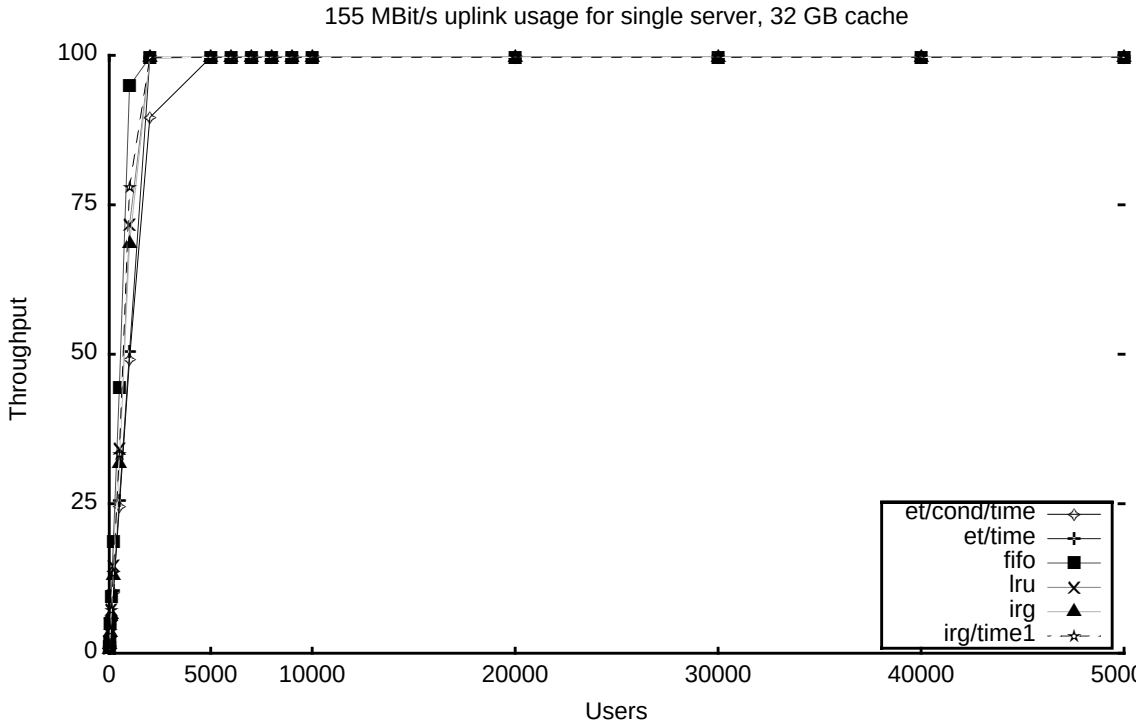


Figure 80: Throughput development with increasing number of users, 155 MBit/s uplink, 32 GB cache

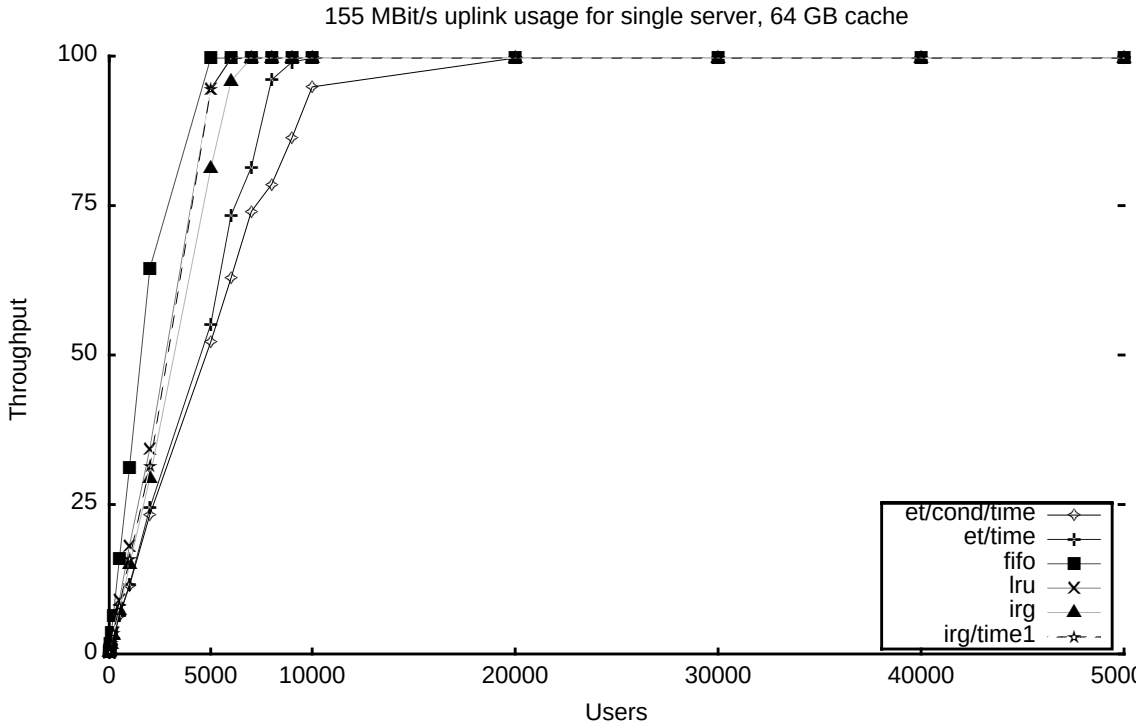


Figure 81: Throughput development with increasing number of users, 155 MBit/s uplink, 64 GB cache

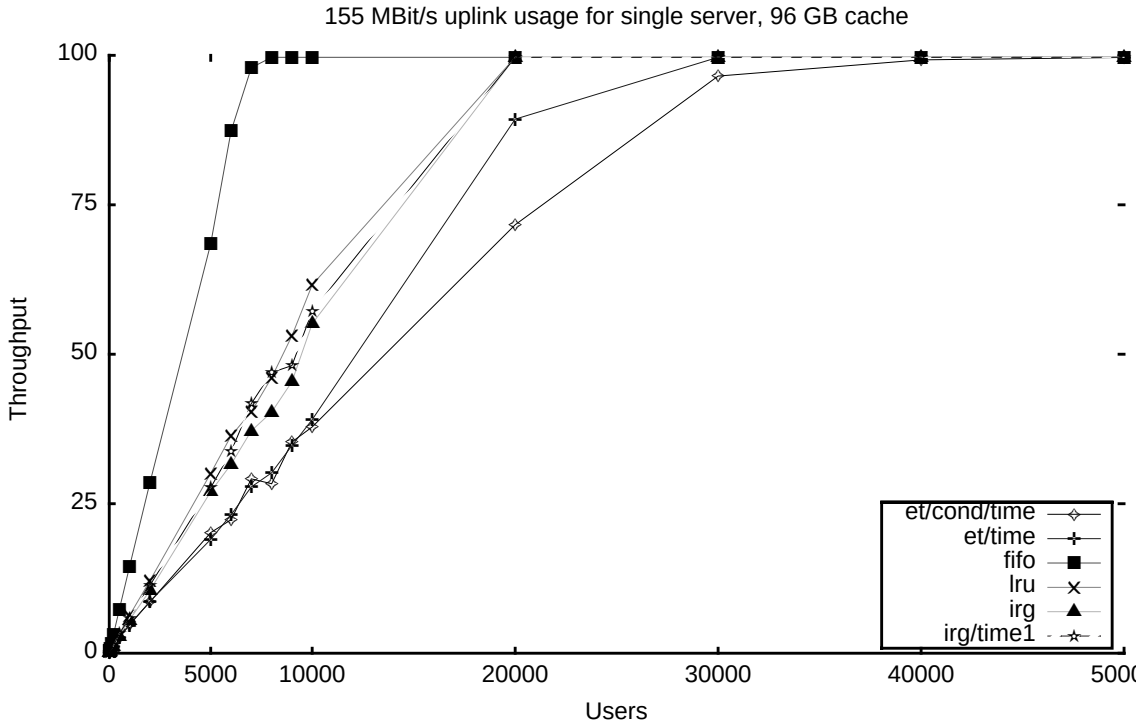


Figure 82: Throughput development with increasing number of users, 155 MBit/s uplink, 96 GB cache

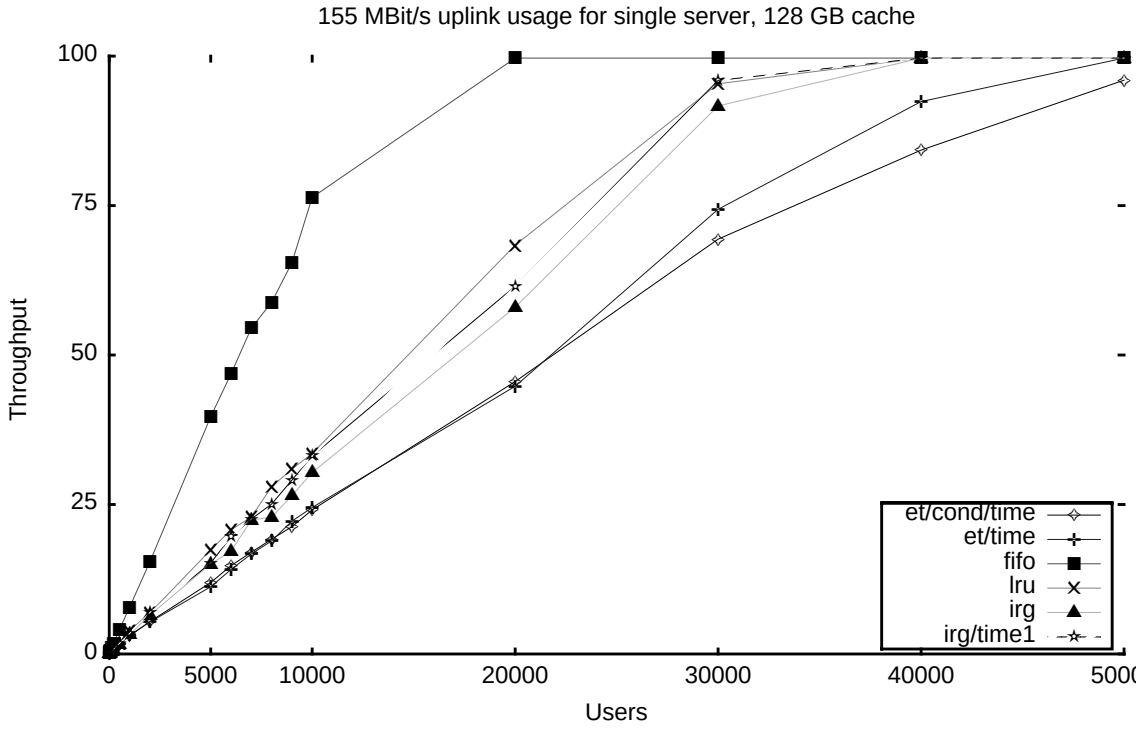


Figure 83: Throughput development with increasing number of users, 155 MBit/s uplink, 128 GB cache

11. Abbreviations

ACK	Acknowledgement
AIX	Advanced Interactive Executive
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
ASF	Advanced Streaming Format
ATM	Asynchronous Transfer Mode
AVP	RTP Profile for Audio and Video
B-frame	Bi-directional coded frame
CD	Compact Disk
CLUT	Color Lookup Table
CNCL	ComNets Class Library
CPU	Central Processing Unit
CSRC	Contribution Source
DAVIC	Digital Audiovisual Council
DCT	Discrete Cosine Transformation
DES	Data Encryption Standard
DHCP	Dynamic Host Configuration Protocol
DiffServ	Differentiated Services
DSM-CC	Distributed Storage Media Command and Control
DVB	Digital Video Broadcasting
ECT	eternal history, conditional overwrite and temporal gap
ETSI	European Telecommunications Standardization Institute
EU	European Union
FCFS	First-come-first-serve
FEC	Forward Error Correction
FIFO	First-on-first-out
FTP	File Transfer Protocol
GB	Gigabyte
GoP	Group of Pictures
HDTV	High Definition Television
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IBM	International Business Machines
I-frame	Intra-coded frame
I/O	Input / Output
IP	Internet Protocol
IPv6	Internet Protocol Version 6
IDEA	International Data Encryption Algorithm
IDL	Interface Definition Language
IntServ	Integrated Services

IRG	Inter-Reference Gap
JPEG	Joined Photographic Pictures Experts Group
Kb, KB, Kb/s	Kilobit, Kilobyte, Kilobit per second
LAN	Local Area Network
LFU	Least frequently used
LRD	Lowest reference density
LRMP	Leight-weight Reliable Multicast Protocol
LRU	least recently used
Mb, MB, Mb/s	Megabit, Megabyte, Megabit per second
MBone	Multicast Backbone
MHEG	Multimedia/Hypermedia Information Coding Experts Group
MPEG	Motion Pictures Experts Group
MMUSIC	Multiparty Multimedia Session Control
NTSC	National Television System Committee
NVoD	Near Video-on-Demand
P-frame	Prediction-coded frame
PAL	Phase Alternating Line
QoS	Quality of Service
RFC	Request For Comments
RMTP	Reliable Multicast Transport Protocol
RSVP	ReSerVation Protocol
RTCP	RTP Control Protocol
RTP	Real-time Transfer Protocol
RTSP	Real-Time Streaming Protocol
RTT	Round-trip-time
SAP	Service Announcement Protocol, also Service Access Point
SDP	Session Description Protocol
SEC-MPEG	Secure MPEG
SL	Server Layer
SLP	Service Location Protocol
SMIL	Synchronized Multimedia Integration Language
SRM	Scalable Reliable Multicast
SSRC	Synchronization Source
ST-II	Stream Protocol Version 2
SUN	Stanford University Network
TCP	Transmission Control Protocol
TRM	Transport Protocol for Reliable Multicast
TV	Television
UDP	User Datagram Protocol
VBR	Variable Bit-rate

VCR	Video Cassette Recorder
VEA	Video Encryption Algorithm
VoD	Video-on-Demand
WAN	Wide Area Network
W3C	World Wide Web Consortium
WWW	World Wide Web
XML	Extensible Markup Language
XOR	Exclusive OR
XTP	eXpress Transfer Protocol

12. Trademarks

AIX, IBM, PowerPC, RS/6000 and VideoCharger are trademarks or registered trademarks of IBM Corporation

Apple, Macintosh and QuickTime are trademarks or registered trademarks of Apple Computer, Inc.

Linux is a trademark of Linus Torvalds

Microsoft, Windows and Windows NT are trademarks or registered trademarks of Microsoft Corporation

RealNetworks, RealSystem, RealVideo, Real G2 with Flash, RealPlayer, RealServer, RealProxy, RealProxy Cache, Basic Server Plus, and SureStream are trademarks or registered trademarks of Real Systems, Inc.

Solaris, Java and Java-related products are trademarks or registered trademarks of Sun Microsystems, Inc.

ViVo ViVoActive are trademarks or registered trademarks of Vivo Software, Inc.

Possible Trademarks

ActiveMovie may be a trademark or registered trademark of Microsoft Corporation

ClearVideo may be a trademark or registered trademark of Iterated Systems, Inc.

MpegTV may be a trademark or registered trademark of MpegTV LLC.

Tabellarischer Lebenslauf

NAME: Carsten Griwodz

Persönliche Daten

Geburtsdatum:	28. Juni 1968
Geburtsort:	Gadderbaum (Bielefeld), Deutschland
Familienstand:	ledig
Staatsangeh.:	deutsch

Anschrift (privat)

Schepp Allee 47
64293 Darmstadt

Tel.: +49 6151 319026

Anschrift (beruflich)

Industrielle Prozeß- und Systemkommunikation (KOM)
Darmstadt University of Technology
Merckstraße 25
64285 Darmstadt

Tel: +49 6151 166159
Fax: +49 6151 166152
EMail: carsten.griwodz@kom.tu-darmstadt.de

berufl. Werdegang

Datum	Berufsbezeichnung	Abteilung	Institution
1997-99	wissenschaftl. Mitarbeiter	Fachbereich Elektrotechnik und Informationstechnik	TU Darmstadt
1997-98	Hospitant & Projektmanager	European Networking Center Heidelberg	IBM Deutschland
1995-97	Wissenschaftl. Berater	European Networking Center Heidelberg	IBM Deutschland
1993-95	Gastwissenschaftler	European Networking Center Heidelberg	IBM Deutschland

Ausbildung und Wehrdienst

1988-93	Diplom-Informatiker, FB17 Informatik, UGH Paderborn
1987-88	Wehrdienst, Deutsche Bundeswehr, Augustdorf
1978-87	Abitur, Kreisgymnasium Halle, Halle/Westf.

Sprachen:

Deutsch (Muttersprache), Englisch (fließend), Französisch (Grundkenntnisse)

Standardisierungsaktivitäten

1995-97	Mitglied im DIN (Deutsches Institut für Normung)
1995-97	DIN Repräsentant bei ISO/IEC JTC1/SC13/WG12 (MHEG)

Lehrtätigkeiten

WS 1997/98	Seminar	“Hochgeschwindigkeitsnetze und Video-on-Demand”
SS 1998	Seminar	“Verteilte Dateisysteme und Netzwerkdateisysteme”
WS 1998/99	Seminar	“IP Multicast und das MBone”
SS 1999	Seminar	“Verteilte Dateisysteme und Netzwerkdateisysteme”

Auszeichnungen

1999	SUR-Grant	“Shared University Research Grant” von IBM

Patentschriften

Carsten Griwodz and Jörg Winckler, Objects With Self-Reflecting Object Relevance Functions, Einreichung bei europ. Patentamt GE998054 durch IBM 1998, Einreichung bei US Patentamt durch IBM 2000

Zeitschriftenbeiträge

Lars C. Wolf, Carsten Griwodz, Ralf Steinmetz, “Multimedia Communication”, Proceedings of the IEEE, Invited Paper in Special Issue on Global Information Infrastructure, Vol. 85, No.12, December 1997, pp. 1915-1933

Konferenzbeiträge

Ralf Steinmetz, Carsten Griwodz, Michael Liepert, Giwon On, Michael Zink, “Consistency QoS”, Networking 2000, May 14-18, 2000, Paris, Frankreich

Michael Zink, Alex Jonas, Carsten Griwodz, Ralf Steinmetz, “LC-RTP (Loss Collection RTP): Reliability for Video Caching in the Internet”, NGITA’00, July 4-7, 2000, Iwate, Japan

Carsten Griwodz, Michael Zink, Michael Liepert, Giwon On, Ralf Steinmetz, “Multicast for Savings in Cache-based Video Distribution”, Multimedia Computing and Networking 2000, January 25-27, 2000, San Jose, CA, USA

Carsten Griwodz, Michael Zink, Michael Liepert, Ralf Steinmetz, “Position Paper: Internet VoD Cache Server Design”, In ACM Multimedia 99 (Part 2), pages 123-126. ACM, October 1999.

Michael Liepert, Carsten Griwodz, Michael Zink, Ralf Steinmetz, “Maintaining Multimedia Lectures with medianode”, SPIE Photonics East ‘99, September, 1999, Boston, MA, USA

Carsten Griwodz, Michael Liepert, Michael Zink, Ralf Steinmetz, “Tune to Lambda-Patching”, ACM Performance Evaluation Review, 27(4), pp. 20-26, March 2000

Carsten Griwodz, Oliver Merkel, Jana Dittmann, Ralf Steinmetz, “Protecting VoD the Easier Way”, ACM Multimedia 1998, September 12-16, 1998, Bristol, UK.

Carsten Griwodz, “Video Protection by Partial Content Corruption”, Multimedia and Security Workshop at ACM Multimedia 1998, September 12, 1998, Bristol, UK.

Abdulmotaleb El-Saddik, Carsten Griwodz, Ralf Steinmetz, “Exploiting User Behaviour in Prefetching WWW Documents”, International Workshop on Interactive Distributed Multimedia Systems and Telecommunication Services’98, Oslo, Norway Sept. 08-11, 1998

Carsten Griwodz and Ralf Steinmetz, “Life-Cycle Considerations for Wide-Area Distritution of Multimedia Data”, in High-Performance Networks for Multimedia Applications, Kluwer Academic Publishers, 1998. Edtrs. A. Danthine, O. Spaniol, W. Effelsberg, D. Ferrari. ISBN 0-7923-8274-9

Carsten Griwodz, Michael Bär, Lars C. Wolf, “Long-term Movie Popularity Models in Video-on-Demand Systems or The Life of an on-Demand Movie”, ACM Multimedia 1997, November 9-13, 1997, Seattle, WA, USA.

The HyNoDe Consortium, “Personalised News on Demand: The HyNoDe Service”, in Proceedings of the Fourth International Workshop on Interactive Distributed Multimedia Systems and Telecommunication Services, Darmstadt, Germany, September 10-12 1997.

H. Coßmann, C. Griwodz, G. Grassel, M. Pühlhöfer, M. Schreiber, R. Steinmetz, H. Wittig, and L. Wolf, “Interoperable ITV Systems Based on MHEG”, In MMCN, January 1996.

H. Coßmann, C. Griwodz, G. Grassel, M. Pühlhöfer, M. Schreiber, R. Steinmetz, H. Wittig, and L. Wolf, “GLASS: A Distributed MHEG-Based Multimedia System”, In COST 237 Workshop, Kopenhagen, November 20-22 1995.

Carsten Griwodz, Hartmut Wittig, “Intelligent Media Agents for Interactive Television Systems”, In IEEE ICMCS, Washington D.C., May 1995.

Eingeladene Vorträge

Carsten Griwodz, Michael Zink, “Decentralized movie distribution in the Internet”, to be given at DIMACS Workshop on Multimedia Streaming on the Internet, June 12-13, Rutgers University, Piscataway, NJ, USA

Lars Wolf, Carsten Griwodz, “Multimedia Communication”, Talk at COST237 Workshop “From Multimedia Services to Network Services”, December 15-19, 1997, Lisboa, Portugal

Carsten Griwodz, “Interaktives Fernsehen - Ansätze für Multimedia im Wohnzimmer”, Talk at VDI , Regionaler Arbeitskreis Multimedia im Raum Mainz/Wiesbaden, 21.10.97, October, 1997, Mainz, Germany

Technical Reports

Ingo Dahm, Carsten Griwodz, Ralf Steinmetz, “Erläuterungen zu MHEG”, IBM TR 43.9501, IBM ENC, 1995

Carsten Griwodz and Ralf Steinmetz, “Mediaservers”, KOM TR-1998-08, KOM TU Darmstadt, June 1998

Christine Küfner, Carsten Griwodz, “Möglichkeiten für den Einsatz von Lastverteilungsstrategien verteilter Systeme in der Videoverteilung”, KOM TR-1998-07, KOM TU Darmstadt, November 1998

Carsten Griwodz, Michael Zink, Michael Liepert, Giwon On, “Analytical Model of Patching VoD Cache Hierarchies”, KOM TR-1999-03, KOM TU Darmstadt, September 1999

Carsten Griwodz, Alex Jonas, Michael Zink, “Affordable Infrastructure for Stream Playback in the Internet”, KOM TR-1999-07, KOM TU Darmstadt, December 1999

Konferenzorganisation

1997	SPIE/Europto Konferenz als Technischer Programmkoorinator
2000	NGITA’00 als PC Member
2000	SEMA 2000 als PC Member

Betreuung von Forschungsarbeiten

1993 Thorsten Illies Diplomarbeit, FH Heilbronn

Entwurf und Entwicklung einer Anwendung für interaktives Fernsehen basierend auf einer anzupassenden MHEG-Laufzeitumgebung für OS/2

1994 Stephan König Studienarbeit, W.-Informatik, Uni Mannheim

Realisierung eines Objektverwalters zur Auflösung von Referenzen und Verwaltung von MHEG Objekten

1995 Thomas Schmitt Studienarbeit, BA Mannheim

Darstellungsmodul für JPEG-Bilder

1995 Thierry Ferey Institute Eurecom, Sophia Antipolis, Frankreich

Design and Implementation of a Virtual MHEG Object Store for the creation of personalized presentations in MHEG

1996 Torsten Linde Diplomarbeit, Informatik, TU Dresden

Die Basisformate InfoDesigner, MHEG5 und HTML 3.2 und ihre Integration in das neue Open-Layout-Format

1996 Michael Bär Diplomarbeit, W.-Informatik, TU Darmstadt

Verfahren zur Verteilung von multimedialen Inhalten in Netzwerken

1997 Theocharis Lioganos Diplomarbeit, Elektrotechnik, TU Darmstadt

Video-Server- und Multimedia-Filesystem-Architekturen

1998 Oliver Merkel Studienarbeit, Elektrotechnik, TU Darmstadt

Realisierung eines MPEG-1-Players unter dem X11-System in Unix mit Streaming im Internet

1998 Mona Hanafi Diplomarbeit, Elektrotechnik, TU Darmstadt

Markieren und Verfolgen von Objekten in MPEG-1 (basierten) Videodatasträmen

1998 Holger Kurth Diplomarbeit, Elektrotechnik, TU Darmstadt

Analyse von Real-Time Multimedia Datenströmen in LAN/MAN auf Basis von IP

1998 Christine Küfner Studienarbeit, W.-Ingenieur, TU Darmstadt

Mechanismen zur Videoverteilung: Ansätze verteilter Betriebssysteme

1998 Andreas Faißt Studienarbeit, Elektrotechnik, TU Darmstadt

Simulation für die Untersuchung von Mechanismen zur Videoverteilung in hierarchischen Netzwerken

1999 Alex Jonas Studienarbeit, Elektrotechnik, TU Darmstadt

Design und Implementierung eines verläßlichen Multicastprotokolls mit Loss Collections

1999 Knut Haberkant Studienarbeit, Elektrotechnik, TU Darmstadt

Umsetzung von RTP und RTCP zur Einbettung in das Java Media Framework

1999 Jörg Schneider Diplomarbeit, Informatik, TU Darmstadt

Umsetzung des Real-Time Streaming Protocols für einen Webserver

1999 Frank Priester Diplomarbeit, Informatik, TU Darmstadt
Entwurf eines Multimedia-Filesystems für Linux

2000 Thorsten Claus Diplomarbeit, Informatik, TU Darmstadt
Partielle Verschlüsselung von hierarchisch kodierten Videos