Implementation of scalable online video services

Pekka Lund

University of Tampere
Department of Computer Sciences
Computer Science
M.Sc. thesis
Supervisor: Roope Raisamo
May 2008

University of Tampere
Department of Computer Sciences
Computer Science
Pekka Lund: Implementation of scalable online video services
M.Sc. thesis, 81 pages
May 2008

The subject matter of this thesis is online video services and particularly the implementation of such services and the various technologies that are involved. This thesis aims to provide an overview of the most significant technologies and components that are fundamental for the implementation of typical online video services. It will also introduce the Media CMS, which is a platform for building online video services. To some extent the subjects that are covered in this thesis function also as a summary of the topics that needed to be researched for the implementation of the Media CMS. Media CMS has a strong emphasis on extensibility, flexibility, performance, and security, and this thesis describes the architectural decisions and design principles that were employed to attain those goals. This thesis also describes the most significant components of the system and the mechanisms and structures that are used to utilize and combine the technologies and off-the-shelf components that will be described. It will also describe some of the most significant challenges and problems that needed to be solved during the implementation of the system.

Keywords: online video, video services, streaming media, video formats, video codecs, Media CMS

Table of Contents

1. Introduction

Online video has become more and more significant and commonplace content type within the Internet during the recent years. The growth of online video has been described with terms such as web video explosion [1], and online video in its various forms has been named a current or future killer application [2, 3, 4].

One sign of the growing popularity of online video has been the ever growing number of different kinds of online video services ranging from on-demand and live Internet TV services to video sharing sites with user-generated content. Many of these sites also utilize the strengths of the Internet by having a number of features which traditional television for the most part lacks, such as social networking and true interactivity.

Some online video services, such as YouTube [5], have also experienced phenomenal growth and success. YouTube was founded in February 2005, and by the end of the year 2006 it had already become one of the top 10 most popular Internet sites [6], named as the invention of the year by the TIME magazine [7], and sold to Google with a price tag of $1.65bn [8]. Predictions tend to agree that the current growing trend of online video usage is going to continue. A recent Cisco report [9], for instance, states that “YouTube is just the beginning” and predicts three waves of growth for online video. The first of those waves is the currently dominating online video to PC wave, which is expected to be exceeded by the next wave of online video to the TV screen, which is expected to be followed by the growth of video communications.

The rising popularity of online video has also raised some concerns. The growing bandwidth demands of video content have led to concerns ranging from overloading of ISP networks [10] to calls for significant investments on the backbone [11]. Online video may also have significant impact on traditional content producers and delivery chains, and time will tell the effects of the expected Internet and TV convergence.

This thesis intends to provide an overview of Internet video services, describe some fundamental enabling technologies behind such services, and provide a brief comparison of commonly used off-the-shelf video streaming products that are commonly used as components in online video service implementations. It also introduces the Media CMS platform, which is a media delivery platform that can function as the basis of different kinds of Internet audio and video services. This thesis explains the fundamental architectural decisions and design principles of the system and how it takes into account some of the most significant aspects of heavy duty web service platforms, including flexibility, extensibility, scalability, and security. The system has already been proven to work in the real world, as it has been powering a couple of high profile Internet services for a while now.

While audio is certainly an integral part of most video presentations and an important medium in itself, this thesis concentrates mostly on video and gives less thought on audio content. One significant reason for this selection of focus is that, on the technological viewpoint, video is a far more demanding medium, especially due to higher data rates, and therefore of more interest for the implementation of an online media platform like the Media CMS.

The rest of this thesis can be divided into two logical parts. The first part consist of chapters 2 to 6, which contain an overview of online video services and the most significant technologies and components that are involved. The rest of the chapters concentrate on the implementation of the Media CMS and on how it utilizes and combines the building blocks that are described in the first part. The first part can also be seen to contain some auxiliary information without which one can successfully implement an online video service but which is quite essential in actually understanding how such services function under the hood. This auxiliary information can also be very useful in the fine-tuning of some aspects of such services.

Chapter 2 deals with the significance of online video on both economical and technological viewpoints. Chapter 3 gives an overview of the various video service types from a historical perspective and the current state of affairs. It also lists some example services and some of their features. Chapters 4 and 5 describe a couple of important technological areas, which can have a significant effect on the functionalities and video quality of online video services. Chapter 4 describes the most commonly used video formats and the basics of video compression. Chapter 5 is about video delivery options and delivery protocols. Chapter 6 describes the most commonly used off-the-shelf software products for the playback and streaming of video content and how they utilize the technologies that are described in chapters 4 and 5.

Chapter 7 begins the second part with a brief introduction of the Media CMS, including its history and fundamental design principles. It also gives an architectural overview of the system and its most significant components. Chapter 8 describes the data model and the database implementation of the system. Chapter 9 describes how the system provides easy to use and efficient data queries, which are essential for the overall performance of the system. Chapter 10 concludes with a summary.

2. Significance of Internet audio and video

Recent years have witnessed a lot of developments on the online video arena. These developments have been described with terms such as web video explosion or transition from a “text web” to a “video web” [1, 12, 13]. Various forms of online video have also been called as the potential or already existing killer application for current and future Internet [2, 3, 4]. Wired magazine recently listed online and mobile video as one of six trends driving the global economy [14]. The following sections try to describe the significance of online video by a number of aspects.

2.1. Popularity

Recent reports by both The Nielsen Company [15] and Pew Research [16] show that the majority of broadband Internet users in the U.S. watch online video, and comScore reports that nearly 75% of U.S. Internet users watched online video during September 2007 and an average viewer watched 3 hours of online video during that month [17]. Statistics by Tilastokeskus show that in Finland 31% of Internet users have used Internet for listening to radio or watching TV, which is about the same percentage as those who have read blogs or used chats or forums [18]. The lower percentage as compared to U.S. may be at least partially explained by the fact that most Finnish Internet radios were silenced for years by the actions of the copyright organizations and were just recently reopened [19], and on the video front hit TV series are not generally available for free viewing to the extent they are in the U.S.

2.2. Internet traffic

Video sharing sites, especially YouTube, have become extremely popular. As of this writing, it is ranked third in the Alexa traffic ranking among all Internet sites [6]. According to Ellacoya Networks, YouTube alone comprises approximately 20% of all Internet HTTP traffic and nearly 10% of all Internet traffic, based on usage data of approximately one million U.S. broadband users [20]. According to Cisco, online video traffic was responsible for 13% of total global Internet traffic and 18% of consumer Internet traffic in 2007 [9]. Cisco predicts a six-fold increase for online video traffic between 2007 and 2011, and the growth is also expected to continue way beyond that. Research done by Velocix (formerly known as CacheLogic) shows that 65% of peer-to-peer traffic consists of video content [21].

The popularity and growth of Internet video usage has led to worries and debate about whether the Internet infrastructure will be able to cope with the increasing bandwidth demands. These worries have extended from the overloading of ISP networks to calls for significant investments on the backbone [10, 11, 22]. There have also been questions about who will bear the costs of the necessary network upgrades as new popular video services such as BBC's iPlayer heighten bandwidth demands [23]. Even the suitability of the current Internet technologies for video delivery needs has been

seriously questioned. Internet pioneer Lawrence Roberts, for instance, has stated that: "The Internet wasn't designed for people to watch television. I know because I designed it." [24]

It is evident that video is an important driver for faster Internet connections, which can result in new opportunities for content providers and better services for end users. It can also bring major profits for those networking companies that have already anticipated the video driven demand for network upgrades [25].

2.3. Effects for traditional content providers

The growing consumption of online video may mean changes to the consumption habits of other video content, especially broadcast television. The exact effects are still debatable though. The ICM survey for the BBC [13] suggested that in Britain online and mobile viewing is rising, and 43% of those who watched online video said that as a result they watched less broadcast TV. The same survey, however, showed that just 9% of the population watched online video regularly, and hence it is not clear how well this result applies to the whole population. A preliminary paper by Waldfogel [26] examined the viewing habits of Penn University students and found some evidence that online video consumption decreases broadcast TV viewing, but the average reduction was only about 25 minutes while the time spent on online video was 4 hours.

A recent Accenture survey [27] among senior media and entertainment executives found that more than half of the respondents identified user-generated content as one of the biggest challenges for the traditional content providers. However, 68 percent of them also believed that user-generated content is something that their companies will be able to monetize on.

Online video may also cause changes to many areas related to broadcasting such as distribution chains. Some local TV affiliates, for example, have voiced concerns of being cut out from the distribution chain as direct online distribution advances [28].

2.4. Advertising

Internet advertisement revenues have grown steadily for several years and the use of video has experienced especially great growth. The total U.S. online advertisement spendings were $6bn in 2002 and have since continuously grown with yearly rates between 20 and 35 percent [29]. Video advertisement spendings however are expected to have grown by nearly 90% in 2007 which includes in-page and streaming video ads [30]. This combined percentage still measures only about 4% of total online ad spendings, but eMarketer predicts that video ads will get a share of 11.5% in 2010 [31].

3. Internet video services

While Internet can be though of as simply another distribution channel for video content, it has some significant benefits as compared to most other distribution means. For starters, Internet has a global reach, and it is an inherently interactive medium. These features enable a number of business and interaction models that haven't or couldn't be utilized with other mediums. The following sections will shortly describe some important Internet video services and service types, as well as some noteworthy historical milestones in Internet video delivery.

3.1. Historical perspective

Video content has had some presence in the Internet for quite a while now, and early commercial Internet broadcasts of such content date back to the early days of the WWW. However, the limitations of networks and hardware have significantly slowed down the development and widespread adoption of video services throughout the years. Now the technology has evidently reached a point where good quality online video is finally a reality. A brief look back into the history demonstrates the dependency between video services and resource constraints.

3.1.1. Dependency on networks and hardware

Back in 1998 Jakob Nielsen proposed the Nielsen's Law of Internet bandwidth which states that "a high-end user's connection speed grows by 50% per year" [32]. Nielsen demonstrated his law with a curve that began in 1984 with a 300 bps modem and fit closely to his empirical data over the years. Nielsen also stated that with this growth rate, bandwidth would remain the limiting factor as compared to the expected doubling of computer power every 18 months.

Bandwidth has clearly been a major limiting factor over the years as far as video content is concerned, although other hardware capabilities have also caused some limitations, such as on the use of high complexity compression algorithms. A brief look back into the history shows how online video has advanced as network and hardware capabilities have permitted.

It is also noteworthy that the viability of content services doesn't usually depend on the state of the art systems but rather the capabilities of average users or at least those of the high-end users, to whom Nielsen referred to in his law. Otherwise the potential user base is not big enough to support new services.

3.1.2. Historical milestones

Some notable world's firsts in Internet video services date back to early 1990's, at least when more or less professionally created and widely available content is concerned.

One early enabling technology was the Cu-SeeMe video-conferencing client [33]. It was first developed for the Macintosh in 1992 and provided 4-bit grey-scale video with a maximum resolution of 320x240 pixels. Cu-SeeMe was used for both the first 24-hour real-time world-wide Internet radio simulcast by the WXYC student radio station [34] and for the first television program to be broadcast on the Internet by ABC's World News Now [35], both in 1994. At that time typical modem speeds, as well as the bandwidth demands of these early broadcasts, were in the order of a few tens of kilobits.

RealNetworks (then known as Progressive Networks) was the first of the current major streaming solution providers to introduce streaming solutions with the introduction of RealAudio in 1995 and RealVideo in 1997 [36]. In 1998 KCTU-TV became the first commercial TV-station to broadcast 24/7 over the Internet [37]. The broadcast was streamed in the RealVideo format. At the time Nielsen's curve was at about 130kbps, about the same as a two channel ISDN.

Full length movies also began to appear on the Internet as bandwidths permitted. CinemaNow was the first to offer major studio films on a pay-per-view basis in 2002 and download-to-own feature films in 2004 [38]. "This is not a Love Song" was the first film to be released simultaneously on cinemas and online for streaming and downloading in 2003 [39]. These kinds of services were already practical as megabit class network connections were already widely available for home users, as the Nielsen's Law also indicates. In December 2007 JACKASS 2.5 became the first feature-length major studio backed movie to debut free and exclusively in the Internet [40].

3.2. User generated content

The milestones of the previous section concentrated on the more traditional video usage cases for which the Internet can be seen only as a new distribution channel for professionally generated content that is already distributed via another means. What really sets Internet apart among the distribution channels is the ability to embrace interactivity, social networking, and especially user generated content. The following subsections focus on these aspects.

3.2.1. Web 2.0

Web 2.0 is a popular but hard to define buzzword that is associated closely with different kinds of Internet services that typically employ high interactivity, social networking and user generated content. The term itself with a version number can easily lead one to believe that it is based on some specific set of specifications or technologies. That, however, is not the case. It is really more of a conceptual or business term, rather than a technological one.

The term Web 2.0 has its roots in the 2004 Web 2.0 Conference, whose name was brainstormed by Tim O'Reilly and Dale Dougherty [41]. O'Reilly himself has given a compact definition for Web 2.0 as ”the business revolution in the computer industry

caused by the move to the Internet as platform, and an attempt to understand the rules for success on that new platform" [42]. Other attempted definitions have significant variations mostly depending whether the primary viewpoint is, for example, from a business angle, about social aspects or more of a technological one.

It can be also questioned whether the term is meaningful at all. Tim Berners-Lee, known as the inventor of the Web, is one notable critic of the term. He has described Web 2.0 as a "piece of jargon" which "nobody even knows what it means". He has also questioned whether it in fact differs from the so called Web 1.0 in any significant way and pointed out that the web, as originally designed, already had many of the same characteristics that are commonly attributed to Web 2.0. [43]

There are two main areas that can be identified as characteristic for Web 2.0 services. One is collaboration and user generated content and the other the set of typically utilized technologies. These technologies can be divided into those that enable rich user interfaces, such as Ajax and Flash, and to those that provide server connections and service-oriented architectures, such as XML based interfaces and protocols. YouTube, for example, has all these characteristics with user generated video content, contribution means such as commenting and rating of content, Ajax and Flash based user interface components, and even externally available XML interfaces [44].

3.2.2. Social networking sites

Social networking sites are web sites which allow users to create profiles with some degree of publicity, list connections to others (e.g. friend lists), and traverse such lists (their own or those of others) [45]. Social networking features have also been incorporated into sites which did not originally have them or are mainly focused on other purposes and features. YouTube, for instance, is focused on video sharing but also features social networking.

It should be noted, however, that on sites where social networking is not the main function such features may be in use only by a minority of the users. Halvey and Keane [46] have sampled the usage patterns of YouTube and found that most users simply view videos, and only a minority uses the social networking features. However, those who use them seem to do so quite often, and hence such functionality may be quite important and beneficial for some users.

3.2.3. Social media sites

Social media sites are web sites where, in addition to creating social networks, the users will create or contribute, annotate, and evaluate content [47]. YouTube, for instance, contains a number of social networking features, such as users' profile pages with commenting, friend lists, groups, communities, and private messaging. Users can share new content and comment and rate content provided by others. Both videos and users can be interlinked. Videos can be linked by for example through tags, channels, and

playlists. Users can be linked for instance through friend lists and memberships in groups. YouTube provides a number of automatically generated listings of videos which are, for example, the most recent, most viewed or rated the best within the day, week, month or all times. Similar features are commonly found among other video sharing sites.

3.2.4. Online video sharing

Sharing of user generated video content has become increasingly popular as evidenced by, for example, the number of video sharing sites among the top 100 websites in the Alexa traffic rankings, which include YouTube, Dailymotion [48], Megavideo [49], and Veoh [50], among others. YouTube is obviously the most striking example of this popularity. It was founded in February 2005 [51], and according to Alexa, it was already among the top 100 websites early in 2006, reached top 10 at the latter half of the year, and has been constantly among the top 5 in 2007 [6]. YouTube has also received a lot of visibility within mainstream media, and it has, for example, become a place for political debates [52] and controversies with a number of countries temporarily banning access to the site due to some controversial content [53].

3.3. Professionally created content

While video sharing and social networking sites concentrate mostly on amateur content, there are also a growing number of sites providing professional and commercial content from the television and film industry. These sites tend to provide much better image quality as compared to sites such as YouTube, and some of them are really more on the edge when it comes to pushing the limits of the networks. These will be discussed in the following subsections.

3.3.1. Television broadcasters

Television broadcasters have lately been active in introducing new online television services. ABC was one of the forerunners in the U.S. when it implemented a two month trial during May and June of 2006 by providing free, advertisement supported Internet streaming of a number of hit TV shows [54]. The trial turned out to be a success, and ABC reopened its service in September with more shows and support for local advertisements [55]. ABC was also the first network to begin HD streaming in July 2007 for some shows [56]. Technically the HD streaming uses 720p resolution encoded with a maximum bit rate of 2Mbps [57].

Other U.S. networks also provide streaming of full episodes, and one particularly interesting recent development is Hulu [58], which is a joint venture by NBC Universal and News Corp. It contains on-demand streaming content from several content providers in a free, ad-supported basis. Some content will be also distributed via sites such as MSN, MySpace, and Yahoo!. It is currently limited for U.S. viewers only.

BBC's iPlayer [59] is another particularly interesting and highly visible video service. It is currently limited for U.K viewing only, but BBC is working on an international version. It allows viewers to download or stream programs broadcast within the last 7 days, and when downloaded, they are viewable for one month. The downloads are enabled by P2P technology. Currently the iPlayer downloading functions only with Windows XP or Vista and uses Windows Media DRM for content protection. This has caused a lot of criticism and recently BBC released a streaming version of the service with Adobe's Flash technology to provide multi-platform compatibility [60].

Other U.K. broadcasters have similar services, including Channel 4 and ITV, which are also planning a joint venture with the BBC, currently known as Kangaroo, which would gather their offerings to a single on-demand service [61].

Major Finnish broadcasters also have their own VOD and live video services. YLE Areena [62] is by far the largest and gathers the offerings of all of YLE's radio and TV channels in a single service. Available programs are limited by distribution deals with content owners, and YLE has signaled their desire to add as much programming as possible as new deals permit [63]. Some content is limited for Finnish viewers only and YLE has announced plans to limit some content to TV license payers only [64].

MTV3 Netti-TV is a VOD streaming service offering programs from MTV3 and SubTV [65]. Some programs are free and others require payment. MTV3 also offers online video streaming rentals by SF Anytime as MTV3 Anytime. Nelonen also has a VOD streaming service with both free and pay content [66]. Recently they introduced their “Hot from the US” service, which provides access to episodes of a number of US series within a week of their premiere within the US, as compared to a typical presentation delay of 6 to 12 months in Finland.

3.3.2. New television providers

Traditional television broadcasters are certainly not the only ones providing television content or television like content on the Internet. Many recent and notable Internet TV services utilize peer to peer technology (P2PTV) for either downloadable or streamable content. Miro [67] and Vuze [68] are examples of services which provide downloadable content, both of which utilize BitTorrent technology [69] and also feature HD content. Services that provide streamable content can be roughly divided to those that provide on-demand content and to those that provide live content. The former include for example Joost [70] and BabelGum [71], among others. Examples of the latter include TVUPlayer [72] and SopCast [73].

These services feature a wide range of content ranging from major TV networks to specialized channels for niche audiences. There are also a number of shows which are distributed on their dedicated sites instead of content aggregation services. MariposaHD [74], for instance, claims to be the world's first HD TV series distributed directly on the

Internet. It is available in 1080i Full HD resolution as BitTorrent downloads. Koeajo.tv [75] is an example of a TV show which was originally shown on broadcast TV but was moved to the Internet. It hasn't got new episodes for a while though.

3.3.3. IPTV

IPTV (Internet Protocol Television) is another recent development that is changing the television and video landscape. The term itself is often vaguely defined and misused and oftentimes used interchangeably with Internet television. They are however two quite different things by most definitions, even though both refer to video delivery over IP networks. While there is a remarkable variance on the definitions of the term, it is commonly characterized as a telco or ISP operated television service where the video streams are delivered over the private (or semi-closed) IP based networks owned by the same operator and usually transferred to the television via a set-top box. It can be defined for example as “multichannel and on-demand programming delivered via DSL or fiber and typically offered by a telco or broadband service provider”. [76]

IPTV has much in common with other traditional television broadcasting channels such as cable and satellite TV and can be seen simply as a competing delivery mechanism or just as an alternative distribution channel for the same content. However, the use of IP networks brings new opportunities, such as VOD services and interactive content, which can extend IPTV services with features attributed to Internet television.

The major difference between IPTV and Internet television can be seen in the openness of the systems. IPTV systems are typically tightly led and controlled by the operator. They are limited to the operator's network, the operator is on control of what content will be provided, and oftentimes the reception requires a separate set-top box, which can be provided by the operator. Internet television, on the other hand, is typically much more open to any content providers, network operators, geographical areas, and reception devices. There can be of course various limitations such as for which areas the content can be provided by licensing reasons and which software is required for the playback of the content, but the basic infrastructure is not limited in this regard. [77]

It is also noteworthy that IPTV can be provided by the same companies that also distribute television via the cable networks. On the other hand, it can also be a new and significant business opportunity for those operators which do not control other distribution channels.

4. Video formats

Online video can be stored and delivered in a wide variety of commonly used video formats, which can have significant differences in their feature sets and performance characteristics. Basic understanding of these formats, their differences and relations, and their underlying technologies can be of great help in making the right technological choices and in the fine-tuning of various configuration options, such as encoding parameters. The following sections describe the fundamentals of video compression and some of the most popular video formats and codecs.

4.1. Video compression

Efficient video compression is a complex subject, and one can certainly implement online video services without understanding the details of video compression, at least if standard formats and third party streaming and playback components are used. This kind of knowledge can, however, be useful in understanding the balance between video quality and compression ratios and the effects of various encoding options. This section describes some mechanisms of video compression through example video codecs, such as the ITU-T [78] standards H.262 (MPEG-2), H.263, and H.264, which are described in section 4.2.

4.1.1. Rationale for video compression

The need for video compression becomes evident with a few simple calculations. For example, a single uncompressed image file with 720x576 pixel SDTV resolution and 24 bit color depth requires 1.2 megabytes of storage and increasing the resolution to 1920x1080 HDTV requires 5.9 megabytes. While these files are big, they are still manageable, and hence the use of uncompressed image files is not that uncommon. However, an uncompressed video at 25 frames per second would already consume about 30 megabytes (237 megabits) per second for SDTV and 148 megabytes (1.2 gigabits) for HDTV. One minute of video would equal to 1.7 and 8.7 gigabytes, and one hour would be 104 and 521 gigabytes. When these figures are compared to Internet video streaming rates, which are typically well under one megabit, or to the DVD disc size, which is less than 4.4 gigabytes, it is clear that some serious compression is needed. In practice these kinds of compression ratios can only be accomplished with some loss in quality.

Lossy compression is a trade-off between quality, data rate, and the computational cost of encoding and decoding. The recent and highly efficient H.264 video codec, for example, typically has a coding gain of about 25% as compared to the previous standard H.263 high compression profile and about 50% as compared to the earlier H.262 (MPEG-2) standard or H.263 baseline [79]. The price of this added efficiency is typically higher decoding complexity and greater CPU demands. H.264 baseline decoding complexity, for example, is two to three times higher than that of H.263 baseline [80].

4.1.2. Relationship to image compression

Most video compression formats, and especially the MPEG standards, are based on a combination of JPEG-style lossy image compression and motion compensation. Video is fundamentally a sequence of images, but as consecutive frames tend to resemble each other, these temporal similarities can be used to improve compression. Some formats, such as Motion-JPEG (M-JPEG) [81] and MPEG formats compressed with "I-frame only" options, do not utilize the temporal direction, but instead compress the video simply as a sequence of independent images. These formats, however, tend to achieve lower compression ratios than those that utilize the temporal similarities.

4.1.3. Video data divisions and subdivisions

Most video codecs, such as H.264, which is used as an example in this subsection, divide the video data contents in a number of ways. First of all the video stream consist of a sequence of pictures which may be either frames or fields depending on whether the video is interlaced. Interlaced video divides each full frame into two consecutive fields, one of which contains the even-numbered rows and the other the odd-numbered rows of a single frame. Interlacing allows the doubling of the frame rate without a significant increase on the file size, but the image quality is not as good as in progressive (non-interlaced) images as the rows of a single image may not be quite aligned. This is due to the fact that they are captured at slightly different time instants. [82]

Similarly to, for example, the JPEG image compression standard [83], the individual pictures are separated into Y, Cb, and Cr components, of which the latter two can be downsampled by reducing their resolution by a factor of two, both horizontally and vertically, which is likely to cause little loss in the perceived image quality but achieves significant data reduction. [82]

Individual pictures can be further divided into slice groups and slices which consist of macroblocks. Macroblocks are 16x16 pixel areas of the image. Slice groups are used by an optional H.264 feature known as FMO (Flexible Macroblock Ordering). If slice groups are not used, the entire picture can be though of as a single slice group which can be divided into slices. [82]

Slices are for the most part self-contained so that they can be decoded independently on the other slices of the picture (except that the H.264 deblocking filter may need to access blocks from other slices). Slices also specify the allowable types of the contained macroblocks as will be discussed in the next subsection. [82]

4.1.4. Frame and macroblock types

Many video encoding formats, such as the MPEG family of standards, define a number of different frame types of which some are compressed as independent images while others are encoded in reference to one or more reference frames.

MPEG-2 and H.264 formats, for example, support the following three frame types that are listed below in Table 1. H.264 actually defines additional SP and SI frame types, which are collectively known as switching slices, but those are not discussed here [84].

Frame type	Also known as	Description
I-frame	intra-frames, keyframes	Frames which are encoded as individual images and hence are not dependent on other frames.
P-frame	predictive frames	Frames which are predicted based on the previous frames. Basically they are (at least partially) coded as differences and translations of the previous frames.
B-frame	bi-directional frames, bi-predictive frames (H.264 [84])	Frames which are predicted on both previous and future frames. This obviously requires that the future reference frames must be known in advance before B-frames can be decoded. This is accomplished by storing the frames out of sequence so that the B-frames follow all the frames they depend on, and these frames are then re-ordered by the player before playback.

Table 1: MPEG video frame types [82]

I-frames and P-frames are also collectively known as reference frames, while P-frames and B-frames are collectively known as delta frames or inter-frames.

The frame types actually specify the macroblock types that are allowed within such frames (I-, P-, and B-macroblocks). In the H.264 standard these types are actually defined per slice (allowed macroblock types within a slice). I-frames or I-slices can only contain intra macroblocks, whereas P-frames or P-slices can contain both intra and predicted macroblocks, and B-frames or B-slices can additionally contain bi-directionally predicted macroblocks. The encodings of different macroblock types are described in subsection 4.1.6. [82]

4.1.5. Groups of Pictures

MPEG video stream is composed of a series of Groups of Pictures (GOP), which are sequences of different types of frames. A GOP always contains a single I-Frame as the first frame which is then followed by any number of P- and B-frames. A typical GOP sequence is IBBPBBPBBP..., which contains a couple of B-frames between the P-frames. Typically one GOP contains something in the order of 5 to 30 P-frames. [85]

Video encoders may provide a number of options for controlling the length of a GOP and the ratio of different frame types. Encoders may also be able to automatically detect scene changes and begin a new GOP there, as the next frame cannot be effectively predicted from the previous frames. These encoder options can have significant effects on the compression ratios and the resulting video and streaming quality. I-frames take more space than P-frames, and B-frames take typically even less space. However, the number of B-frames between I- and P-frames should be limited to a few frames as B-frame prediction is usually based only on the surrounding P- and I-frames (H.264 also allows prediction based on B-frames [82]), and the use of too many consecutive B-

frames decreases the correlation between them. Wu *et al*. [85] suggest that the number of consecutive B-frames should be close to two, as their experiments show that larger numbers will not result significant reductions in the file size (and may actually increase it), and they will quickly reduce image quality. Their results also show that increasing the number of P-frames to more than five does not yield significant gains in terms of file size, and the image quality stays pretty much the same, and hence they recommend a maximum of five P-frames per GOP.

The number of P-frames, or rather the interval between I-frames, may not be that important in downloaded content, but it is very significant in streaming use. This is because streaming can only begin on a keyframe (I-frame), which is independent of other frames. Fast forwarding and rewinding positions, for example, can be targeted only to the nearest keyframe, which are already more than a second apart if the frame rate is 25 frames per second and there are 10 P-frames in a GOP with 2 B-frames between each of them. It also means that if a keyframe, or part of it, is lost because of a network error, then it may take more than a second before the affected image area will be fully fixed with a new keyframe. Lost image parts can, however, be retransmitted and specifically this problem does not exist if streaming is performed using a reliable protocol, such as TCP. It is noteworthy that the choices in compression parameters such as the GOP length and the selected usage types and transport protocols are interdependent, and this should be taken into account when optimizing those parameters.

4.1.6. Motion compensation

Motion compensation utilizes temporal redundancy by encoding image areas as differences (or residuals) of similar areas within reference pictures. Consecutive video frames are likely to contain very similar image areas in slightly altered places, as either the objects or the camera may be moving between the frames. Many video encoding formats use motion estimation for finding those reference areas and motion compensation for utilizing those findings by encoding macroblocks as differences to the reference areas found by motion estimation. [82]

H.264, for example, allows the use of multiple reference frames for the prediction of one frame. These reference frames can be either past or future frames in time order, but obviously the use of future reference frames requires that they are decoded before the referencing frames. The use of multiple reference frames also multiplies the number of necessary image buffers. Prediction is done on the level of 16x16 macroblocks or their subdivisions down to 4x4 blocks. These blocks are predicted by a reference area which is referred to by a combination of reference frame index and motion vector. The motion vector specifies the relative location of the reference area within the reference frame and it can be specified with a quarter pixel accuracy. This sub-pixel accuracy yields better compression performance but also requires that non-integral sample values are interpolated, which increases computational complexity. [82]

Each P-frame, or more precisely each P-slice in H.264, can encode each macroblock as either an intra or inter block. Intra blocks can be predicted only by replicating or averaging the values of the neighboring pixels within the same frame. Inter blocks can be predicted as a motion vector from a single reference frame. Different blocks of a single slice can use different reference frames. B-frames add another type of prediction where a block can be predicted as a weighted average of two different reference frames and motion vectors. In all of these predicted cases, the block is encoded as a residual difference from the reference pixels. H.264 also allows blocks to be encoded without any prediction, in case there are no suitable reference areas and even intra-prediction based on the neighboring pixels would only cause data expansion. [82]

4.1.7. Bit rate options

Compression performance is greatly affected by the complexity of video contents. Complexity in turn is dependent on the amount of detail and motion, both of which can have a lot of variance, even within a single video clip.

Video can be compressed with either constant bit rate (CBR) or variable bit rate (VBR) encoding [86]. CBR keeps the bit rate constant, or at least relatively constant, regardless of the complexity of the current video segment. VBR, on the other hand, can have a lot of variance on the bit rate by using only minimal amount of bits for segments without action or much detail and a lot more on the complex segments. This way VBR can be used to attain a constant quality regardless of the changes in complexity while avoiding the waste of bits when they are not needed. [87]

The bit rate range of VBR encoded video is likely to be bound with limits on the minimum and maximum bit rates. The overall file size can be controlled by specifying the average bit rate for the entire video, in which case the encoder distributes the available bits as it sees fit within those limits. The video can also be encoded with a specific target quality, in which case the resulting file size will be less predictable. In addition to the overall limits, there can also be limits on the average bit rate within some time interval. This can be necessary to limit the peak resource consumption in long complex segments. Network bandwidth in video streaming is one such scarce resource for which bandwidth fluctuations and peak bit rates need to be limited. [82]

VBR encoded video usually achieves better overall quality than a CBR encoded video of the same file size as the available bits are distributed more effectively. VBR encoding can also have the advantage of being simpler on the encoder side, as CBR encoding needs to adjust the quantization parameters dynamically to keep the bit rate constant regardless of the complexity of the video. VBR video, on the other hand, can keep those parameters constant. However, VBR encoding may need to adjust those parameters as well if it is constrained by the limits on the peak bit rates. [82, 87]

VBR video can also provide statistical multiplexing gains on streaming as it has a lower average bit rate with the same quality and hence a streaming server could serve a larger amount of clients with the same network bandwidth. However, due to the bit rate variance, the resource consumption is unpredictable and can have worst case scenarios with high bandwidth peaks and server resource exhaustion. Live streaming via unicast is especially problematic for VBR encoding, as the bit rate peaks happen at the same time for all the client streams. These problems can be reduced with online bandwidth smoothing techniques which use buffering for transmitting the otherwise bursty traffic with a smoother rate [88].

Many encoders also provide the choice between 1-pass and 2-pass encoding. The latter one performs two phases over the video data by first analyzing the video for selecting the optimal encoding parameters for the actual encoding in the second phase. This can result in significant compression gains with the expense of slower encoding. 2-pass encoding is mostly used with VBR encoding where it can be used to optimize the bit rates over longer periods of time among other parameters. It can also be an option with CBR encoding, in which case the optimization is more limited. [82]

The exact compression gains of VBR over CBR depend on a number of factors, including the contents and complexity of the video and the codec and encoding parameters that are used. In some cases VBR encoded files can be only half of the size of an equal quality CBR encoded file [86]. Koumaras *et al.* [87] compared the perceived quality of CBR and VBR encoded MPEG-4 ASP clips with CIF and QCIF resolutions and found a quality difference of approximately 4-5% for CIF resolution content and around 2.5% for QCIF content as measured by the combined result of four different objective metrics. It is, however, unclear how well these results translate to other codecs and higher resolutions, and the use of some significant parameters such as 2-pass encoding is not explicitly stated in their work (the encoder preset setting of “High Quality” is mentioned).

4.1.8. Multiple bit rate encoding

Multiple bit rate (MBR) encoding means that the same video content is encoded with a number of different bit rates which are placed within the same file as alternative versions [89]. Each of those is encoded separately with either CBR or VBR encoding and is independent of the others. The advantage is that the bit rate can be selected depending on the capabilities of the client and the available network bandwidth in case the file is streamed. The file size will be approximately the combined size of the different bit rate versions, but in the streaming case only the selected version needs to be streamed to the client. Many streaming servers and clients are also capable of switching between versions on the fly if the available network bandwidth changes.

4.2. Codecs

Video codecs are the (hardware or software) components responsible for the compression and/or decompression of the actual video data contents. There are plenty of codecs to choose from, and the following subsections list some of the most notable ones.

4.2.1. ITU-T and ISO/IEC standards

The standardization of video encodings can be mostly attributed to two standardization groups, namely the Moving Picture Experts Group (MPEG) of ISO/IEC [90] and the Video Coding Experts Group (VCEG) of ITU-T [78]. These groups have worked together on many occasions and have also published some common standards, which are technically identical but are named differently by those groups. Hence MPEG-2, for example, is known as ITU-T H.262 and also with the formal ISO/IEC standard number ISO/IEC 13818. To further confuse things, these standards typically consist of multiple parts, each of which has its own naming conventions within their respective organizations, and oftentimes they have both a formal name and another more descriptive name. Hence the H.264 video compression standard is also known as MPEG-4 Part 10, MPEG-4 AVC (Advanced Video Coding) and ISO/IEC 14496-10 as well as with combined names such as H.264/AVC. [91]

The first digital video coding standard was H.120, published in 1984 by ITU-T (then known as CCITT), but the video quality didn't prove to be good enough for widespread deployment [92]. H.261, published in 1990, was the first practical and successful standard. It operated at 64kbps-2Mbps and already contained many of the features which are still in use in current standards including macroblocks, DCT, scalar quantization, zigzag scan, and RLE. [91]

MPEG-1 was standardized in 1993 and provided approximately VHS quality with bit rates between 1-2Mbps. It is still in use in the video CD (VCD) format and especially the MPEG-1 Audio layer III, better known as MP3, is a popular audio format. [91]

H.262, alias MPEG-2/Video (ISO/IEC 13818-2), which was developed in 1994, was the first joint video standard by the two standards bodies. It is similar to MPEG-1 with a few additions, such as support for interlaced video. It is used in SDTV broadcasts, such as the first generation DVB, with typical bit rates of 2-5Mbps, in DVD format with 6-8Mbps, and in HDTV broadcasts with about 20Mbps. [91]

H.263 was the next video codec standard by ITU-T. The original version was ratified in 1996 and it was followed by improved versions in 1998 and 2000. Those are known as H.263+ and H.263++, respectively [93]. The H.263 baseline core mode is interoperable with MPEG-4 part 2 ASP (Advanced Simple Profile) [94].

H.264 also known as MPEG-4 part 10, MPEG-4 AVC (Advanced Video Coding), and ISO/IEC 14496-10 is the newest joint standard by ITU-T and ISO/IEC. MPEG-4 AVC should not be confused to the entire MPEG-4 suite of standards, which also

includes another notable video codec as part 2, also known as MPEG-4 Visual, and often referred simply and rather confusingly as just MPEG-4. It is similar to MPEG-2/Video with a performance between MPEG-2/Video and MPEG-4 AVC. H.264 achieves a superior performance as compared to the previous standards and commonly achieves the same picture quality at half the bit rate or less as compared to MPEG-2 [95, 96]. Some of most significant differences as compared to the previous standards are the hybrid predictive/transform intra frame coding, in-loop deblocking filter, and the use of 4x4 integer transformations instead of the typical 8x8 DCT. [82]

H.264 has quickly gained popularity, and it is adopted widely for many uses [97]. It is a mandatory feature of the next generation DVD formats HD DVD and Blu-ray Disc and already used or selected for use in many digital television broadcasting systems, especially with HDTV resolutions, where the significant bandwidth gains as compared to MPEG-2 are essential. It is also supported by the latest versions of the QuickTime and Flash formats and used by many Internet video services including P2P services Joost [70] and BabelGum [71].

4.2.2. Other notable codecs

VC-1 is the informal name of a video codec standardized by the Society of Motion Picture and Television Engineers (SMPTE) as SMPTE 421M [98]. VC-1 is based on Windows Media Video 9 codec (also known as WMV3 by its FourCC code), with which it is functionally identical, and hence WMV 9 is Microsoft's implementation of the VC-1 standard. VC-1 is a DCT based codec with many similarities to H.264. It is a mandatory codec for both HD DVD and Blu-ray Disc formats, alongside H.264 and MPEG-2. WMV 9 supports constant bit rate (CBR) and variable bit rate (VBR) encodings, both with one-pass and two-pass options [86].

RealNetworks RealVideo [99] is a pioneering proprietary video codec, especially when it comes to Internet video streaming. The first versions date back more than a decade and it is currently at version 10, which is an encoder side improvement and backwards compatible with version 9 decoders. RealVideo supports CBR and VBR encodings, two-pass encoding, interlacing, and HDTV resolutions.

On2 Technologies TrueMotion codecs are another notable and widely deployed proprietary DCT based codec family [100, 101, 102]. TrueMotion VP6 is supported by Adobe Flash player 8 and above [103], and the latest TrueMotion VP7 is used on services such as Skype video calling [104] and ABC's HD Streaming service [57]. On2 has released an earlier VP3 codec into the open-source community, and it has been used as the basis for the free Theora codec [105].

Dirac [106] is a wavelet based open source video compression family developed by the BBC. Dirac is still under development and its compression performance is expected to be within the same class as that of H.264 and VC-1. Measurements from development

versions have shown performance between MPEG-4 Visual and H.264 as measured by PSTN [107]. The Dirac family of codecs also includes Dirac Pro, which is targeted for high quality production use with bit rates over 100Mbps to more than 1Gbps. BBC aims to standardize it with SMPTE as VC-2.

4.3. Container formats

Most video files are not simply single streams of video data as they contain several different types of data which may be encoded with different codecs and data formats. Different data types are usually contained in a single file as different tracks or streams. Possible track types include the following [108]:

- Video streams
- Audio streams
- Timed text streams (subtitling, closed captioning)
- Still image streams
- Script streams (interactivity, events)
- Web streams (for displaying web content)
- Metadata tags (such as chapter points)

Tracks are typically interleaved so that tracks that are supposed to be played back at the same time are placed close to each other in the file order. There may be several options how the tracks are played as some of them may be always played simultaneously while others may be mutually exclusive or optional.

The file formats that are used to store these collections of tracks are called container formats. Some of them are very generic in nature and place few restrictions on the codecs that can be used, while others may be limited to a few specific codecs. The following subsections list some of the most notable container formats.

4.3.1. Standards

The MPEG-2 standard part-1 (ISO/IEC 13818-1: Systems / ITU-T Rec. H.222.0) [109] defines two separate container formats, which are called the Transport Stream and the Program Stream. The Transport Stream is designed to be used in environments where errors are likely, such as digital television transmissions over the air. The Program Stream is intended for use in relatively error-free environments, such as storage on disc. It is forward compatible to the MPEG-1 systems layer (ISO/IEC 11172-1: Systems), which only defines one container format. Both streams contain the actual elementary data streams, such as video and audio tracks, as PES (Packetized Elementary Stream) packets. Each PES packet contains data from only one elementary stream. The packet headers contain presentation time stamps which are used for synchronization among the different elementary streams.

MPEG-4 defines several file formats in parts 11, 14 and 15 [110]. Additionally a common ISO Base Media File Format is defined as part 12, which is based on the QuickTime format [111]. Part 11 is not a traditional container format but deals with scene descriptions, which may contain audio and video contents but also various other content types. This part defines a textual format (XMT, Extensible MPEG-4 Textual format) which contains features from VRML, SMIL, X3D, SVG, and MPEG-7. Part 14 defines the mp4 container format and part 15 defines a format for storing AVC (part 10) content, with support for additional AVC features. Both are based on the part 12 base format. The 3GPP file format (3GP) [112] is also based on the ISO Base Media File format.

4.3.2. Proprietary formats

ASF (Advanced Systems Format) [108] is a proprietary media container format by Microsoft. It is usually used with Windows Media Audio and Video encoded contents (with .WMA and .WMV extensions, respectively), although the format itself is not limited for Windows Media content types. A single ASF file may contain a number streams which can be grouped as stream groups. Streams or stream groups may be mutually exclusive, which is typically used for storing the same content with alternative bit rates or language versions.

AVI (Audio Video Interleave) [113] is another older but still widely used container format by Microsoft. It is based on the RIFF format (Resource Interchange File Format) and can contain a number of audio and video streams encoded with various codecs.

RealMedia File Format (RMFF) [114] is a proprietary RealNetworks' container format for RealAudio and RealVideo files. It can contain a number of media streams which can be interleaved. Individual physical media streams can be grouped as logical streams which may, for example, contain alternative bit rate versions.

Flash video (FLV) [115] is a proprietary video format supported by the Adobe Flash Player 7 and later versions. It supports only a very limited set of codecs which are detailed in subsection 6.3.5.

QuickTime Movie (MOV) file format [81] consists of a number of tracks which may be media streams, specialized tracks such as interactive sprite tracks or references to streams which are located in separate files. The QuickTime format has many features which make it suitable for editing, and it was selected as the basis for the ISO Base Media Format and is hence very similar to the MPEG-4 container format.

4.3.3. Open source formats

Matroska [116] and Ogg [117] are both open source container formats. Both support multiple streams, streaming, and random access (indexing). Ogg is a simple stream oriented format with low overhead (1-2%), and it is most notably used with Ogg Vorbis

audio. Matroska is targeted to be a standard multipurpose container format with a focus on future extensibility. A major component of this extensibility is that the file format is based on EBML (Extensible Binary Meta Language) [118], which is a binary derivative of XML.

5. Video delivery options

Internet video delivery demands a lot of network bandwidth, and some applications, such as video conferencing, also require low network delays. The selections of delivery methods and network protocols may have significant effects on the overall quality of the video viewing experience, as networks tend to be the most significant bottlenecks in this respect. These selections can have effects on at least the following areas:

- Video quality
- Playback quality (buffering pauses, jitter)
- Playback delays (initial playback delay or delay after fast forwarding)
- Playback functionalities (e.g. pausing, fast forward, off-line viewing)
- Scalability
- Delivery costs
- Effects on other network traffic
- Compatibility with firewalls and NATs

The following sections describe some of the choices in delivery methods and protocols.

5.1. Delivery methods

Online video delivery methods can be divided to the following groups which will be discussed in the following subsections:

- Streaming
- Downloading
- Progressive downloading, also known as pseudo-streaming

While peer-to-peer delivery methods have been used successfully in several video services and have a number of advantages, this chapter focuses only on the traditional client-server models.

5.1.1. Downloading

In this context downloading is meant to signify that a video file is first fully transferred into the user's computer and then played back. This might be the only viable solution if the bit rate of a video file exceeds the available network bandwidth considerably or the file should be playable offline. Downloading can be provided with regular HTTP or FTP servers and protocols, or for example with BitTorrent, and hence it is cost-effective and easy to implement. The downside is that the user must first wait for the entire file to be transferred before the playback begins. Access restrictions can be implemented with DRM systems if needed, as otherwise the downloaded content can be freely copied.

Several VOD services use downloading so that the entire file must be retrieved before the playback begins within the software. These include Miro and Vuze, both of which utilize BitTorrent for the file transfers (Miro can also use simple HTTP).

5.1.2. Progressive downloading

On the network level progressive downloading, also known as pseudo-streaming, functions like regular downloading. The main differences are that the playback starts immediately or after a short buffering delay and random access to the file can be enabled with a proper combination of client, server, and file format features. [119]

Typically progressive downloading with random access requires some support from both the client and the server as well as from the file format. After fast forwarding or rewinding the playback must begin from the beginning of a keyframe. This can be accomplished by injecting the locations of the keyframes as metadata somewhere near the beginning of the file. The client uses this information for requesting the correct byte position from the server, and the server must support requests to a specific byte offset.

Flash video, for instance, can be pseudo-streamed with random access capabilities by injecting video files with a tool, such as Buraks FLV MetaData Injector [120], and serving these files from a web server with the help of a server-side script, such as xmoov-php [121] or a web server extension. The client will read the injected metadata and request specific offsets with the help of an HTTP parameter. YouTube is an example of a service which uses progressive downloading of Flash videos with random access capabilities.

For the most part, progressive downloading shares the strengths and weaknesses of regular downloading. Progressive downloading can, however, provide some, although limited, content protection, as files can be played without ever storing them to the client computer as files, which are easily copied. This however requires that the player implementation doesn't cache content as files on playback, and that the server performs some access checks, such as the exchange of security tokens that prevent direct access to files with unauthorized client software.

5.1.3. Streaming

Streaming enables real-time playback of on-demand or live content with the help of specialized streaming servers and protocols [119]. It has a number of advantages over the other alternatives. It can use multicast and provide live content, both of which are hardly feasible with downloading or progressive downloading. The playback begins almost immediately after a short buffering delay, and no files are transferred to the client computer, which is an advantage for access control, although the content can still be copied with stream grabbing software. Downsides of streaming include the necessity to use specialized streaming servers, which can be costly, and specialized streaming protocols, which can have problems in traversing firewalls. Most streaming solutions,

however, offer the possibility to tunnel streaming traffic over HTTP which solves those firewall problems.

The biggest advantage of streaming is probably the possibility of bandwidth control and adaptation to various network conditions. There are multiple methods for reacting to network congestion, and not all streaming solutions provide them all. These methods include the following [89]:

- **MBR encoding**. The same content is encoded in several bit rates within the same file and the streamed bit rate version can be switched on the fly as needed.
- **Frame-rate dropping**. Possibly only (some of) the keyframes are delivered.
- **Audio only delivery**. Dropping of video altogether and delivering only audio.

Streaming usually has predictable and near constant bandwidth consumption on the server side, which makes it easier to serve more clients than with progressive downloading. Some streaming solutions, however, have certain optional technologies which complicate or even disable rate control mechanisms. Windows Media streaming, for example, has a feature group known as Fast Streaming which includes technologies that buffer data faster than the normal streaming rate. Some of these will disable rate adaptation techniques [89].

5.2. Routing schemes

There are four routing schemes that are directly supported on the IP layer. These are unicast, broadcast, multicast, and anycast. Broadcast is defined as a separate scheme only in IPv4 [122], as it is simply a special case of multicast in IPv6 [123, 124]. Anycast is poorly suited for the actual data delivery of media content in traditional client-server models and will not be discussed here. Other routing schemes are described in the following subsections with especially streaming use in mind.

5.2.1. Unicast

Unicast is a one-to-one routing scheme, and it is also the most commonly used of the routing schemes. Each unicast connection is used to transfer data between only two endpoints. When unicast streaming is used, each client connection is completely separate and takes its own share of bandwidth, even if all clients are receiving the same identical live stream at the same time. It is obviously quite a waste of bandwidth to stream e.g. 1000 identical streams from the same server. Simultaneous unicast live streams may not be quite identical, however, as they may not be exactly in sync with each other and rate adaptation techniques and retransmits may alter the streams individually depending on the quality of the client connections.

5.2.2. Broadcast

Broadcast is a one-to-all routing scheme where the data packets are sent to all possible receivers within the same broadcast domain. Most routers drop broadcast packets, and therefore broadcast is in practice confined within individual network segments. [125]

Broadcast could be used to deliver streaming media efficiently to all computers within an office LAN, for example. The server needs to send only a single stream which is distributed to all viewers. The downside is that the stream is also sent to those computers within the same LAN which are not interested in receiving it, which unnecessarily consumes their resources. This makes broadcast largely impractical, except in some special cases where all or the vast majority of the computers within the network segment are actually interested receivers. [125]

5.2.3. Multicast

Multicast is a one-to-many routing scheme where the data packets are sent to a group of interested receivers. It conserves bandwidth by sending a single stream which is distributed to all interested parties by replicating the packets on routers in a treelike fashion when the distribution tree branches. [126]

IP Multicast is based on multicast groups which are basically reserved addresses from the IP address space. Hosts can join and leave multicast groups and each host within the group receives all packets send to the group. The groups are only used for reachability purposes and no access protection is provided. [126]

The original IP multicast allows each host member to also send packets to the group. This is known as Any-Source Multicast (ASM). Source-Filtered Multicast (SFM) is an ASM variant which allows filtering of sources on the receiver side so that only those packets that are sent to the group by some specific senders (or alternatively all but the specific senders) are accepted. Source-Specific Multicast (SSM) is a simplified multicast model which allows only a single sender per group. [127]

Multicast would be hugely beneficial for many applications where large amounts of the same content needs to be delivered more or less simultaneously to a large number of receivers. Video conferencing and streaming media, especially live streaming, are examples of such applications. Unfortunately, there are a number of problems regarding the use of multicast. First of all, while multicast is generally available within limited domains such as corporate LANs there is limited support for inter-domain multicast within the Internet. This presumably results in from several other problems including the complexity of deploying and supporting multicast, security related issues, the lack of commercial interest for deployment (at least partly due to complexities in charging for the service), and also in part the lack of demand. Some of the complexities and security issues result from the support of multiple senders within the ASM multicast model. In this regard SSM is a lot simpler model, and while some applications cannot be limited to

a single sender this would certainly be sufficient for regular streaming media scenarios. Unfortunately, the support for inter-domain SSM is even more limited than it is for ASM. [128]

Supporting heterogeneous receivers is another problematic area for multicast in applications such as streaming media. The rate adaptation mechanisms and feedback loops which are used in unicast streaming cannot be directly used in multicast, as the network connections vary between the receivers, but a single stream is delivered to all of them [129]. Some possible solutions are discussed in subsection 5.3.4 in regard to the RTP protocol.

5.3. Network protocols

Different video streaming solutions employ different protocols and oftentimes they provide several protocol alternatives and protocol configuration options. Automatic protocol selection based on the client and network capabilities is also a common feature. The choice of network protocols can be very significant for many reasons, which include the following:

- Some protocols may not actually work at all for all clients, due to the capabilities of the client software and network configurations.
- Different protocols react differently to various network conditions, such as congestion. There are differences in how streaming is affected by other data transfers as well as how streaming affects those other transfers.
- Overhead (and hence the actual attainable useful bandwidth) varies by protocol depending on the size of protocol packets and headers.
- The protocol choice may also affect the price of streaming as different protocols may require different streaming software.

5.3.1. Protocol layers

There are always several protocols involved in video streaming, as there are multiple network layers stacked on top of each other, each utilizing protocols of their own.

The lowest layer that is considered here is the network layer, which uses the Internet Protocol (IP) [122, 123] as far as streaming protocols are considered. As a network layer protocol, IP provides the addressing and routing functionalities. The next layer, the transport layer, is currently dominated by the TCP [130] and UDP [131] protocols. There are also other alternatives such as DCCP [132] and SCTP [133], but at the moment they still lack widespread deployment. Most streaming solutions provide a choice between TCP and UDP on the transport layer, and these protocols have some significant differences and considerations when it comes to streaming which will be discussed in the next subsection.

Streaming is commonly performed with separate data transport and control protocols which operate on top of TCP or UDP. There are several alternatives for these protocols, especially proprietary ones, but the current trend is towards the standard RTSP+RTP protocol pair. These protocols will be described in more detail in sections 5.3.3 and 5.3.4. In fact, of the current streaming products from the major players in the field Adobe Flash is the only one which sticks with its proprietary RTMP protocols, whereas the others all support RTSP+RTP and have deprecated most of the older proprietary protocols, which need to be used only when support for some older client software is a requirement.

Older proprietary protocols include MMS, RDT, and PNA. MMS is a streaming protocol used by older Windows Media players and streaming servers. Support for it was removed with Windows Media Services for Windows Server 2008. The mms protocol prefix is still used and recommended for streaming URLs, though, as it is used for protocol rollover (automatic streaming protocol selection by trying the different alternatives in a certain order. RDT is a proprietary streaming protocol which can be used in place of RTP for streaming Real Media content. Similarly to RTP, it is used in combination with RTSP, and current RealPlayers support both RTSP+RTP and RTSP+RDT combinations. Real Media streaming has also used PNA and MMS protocols. [134]

HTTP is another option on the application layer, but it is mostly used on top of other data transport and application layer protocols as a tunneling protocol, rather than as an alternative to them. Most streaming solutions can use HTTP as a fall-back protocol in case other options fail due to firewalls. One important implication of the use of HTTP is the fact that it is always layered on top of TCP, and thus UDP cannot be used when streaming protocols such as RTSP+RTP are tunneled over HTTP. As a tunneling protocol, HTTP adds overhead only to the connection establishment, as the actual data is normally streamed in a couple of persistent connections.

5.3.2. TCP versus UDP

TCP and UDP have a number of differences which are significant in streaming media. TCP is a connection-oriented protocol which provides a reliable and congestion controlled transport but makes sending rate control difficult. UDP, on the other hand, provides connectionless delivery of datagrams with easier control of sending rate and lower header overhead but lacks both reliability and built-in congestion control. UDP is also more likely to be blocked by firewalls. [130, 131]

Streaming media requires a lot of bandwidth and preferably only small fluctuations on the throughput, at least in some usage scenarios. This makes it likely to both cause and suffer from congestion, and hence congestion control is an especially important issue. TCP's window based congestion control [135], however, is problematic for

streaming media as loss of a single packet halves the sending rate which is also increased rather slowly after such event. Multiple or repeated losses reduce the throughput rapidly and significantly, and the throughput is subject to large fluctuations.

One proposed solution is MultiTCP [136], which uses multiple TCP connections for the data transfer of a single multimedia stream. The basic idea is that if the data transfer is divided into several connections, then a packet loss in one of them halves the data transfer rate of only that part of the transfer, and the total impact to the stream will be much lower. In fact, the total amount of lost throughput that a single lost packet will cause when using just two connections will be only a quarter of what it would be with a single connection. MultiTCP also controls the sending rate of individual connections so that the combined bandwidth will be the same as it would be with a single connection. Measurements have shown that it achieves quite a bit smoother and higher throughput than a single TCP connection during various levels of congestion. It has also the benefit of using normal TCP connections without kernel level modifications, although the sending rate control may not be possible with typical higher level APIs.

It should be noted however, that short-term fluctuations aren't necessarily a problem for many streaming media scenarios, as most on-demand and live streaming applications without specific real-time requirements buffer several seconds worth of data, and therefore longer term throughput may be actually more important than short-term stability. Hence TCP may, in fact, be a good protocol alternative despite its bandwidth fluctuation characteristics. Interactive or more or less real-time streaming applications, such as videoconferencing, are quite a different matter, though, as there can't be much buffering delay, and hence fluctuations will be a major issue.

Chung and Claypool [137] evaluated the TCP friendliness of UDP-based RealVideo streams as an instance of commercial and widely deployed streaming solution and found them to be mostly fair with TCP traffic expect in some resource-constrained conditions. Many streaming solutions, such as RealMedia streaming and Windows media streaming, can also utilize MBR encoding and switch to a lower bit rate version when the connection becomes congested.

Streaming media can also usually tolerate some amount of packet loss which can be even at least partially visually concealed. Hence full reliability is not needed and TCP's reliability may be an overkill. Retransmissions are however commonly used also with UDP based streaming protocols.

5.3.3. RTSP

RTSP (Real-Time Streaming Protocol) [138] is application layer protocol for controlling the delivery of real-time data, such as audio and video streams. RTSP is used only for controlling the delivery of data streams and it does not deliver the actual data contents, which are delivered with the help of other protocols. There is, however, one exception to

this, as the actual data contents can be interleaved within the RTSP as for example RTP+RTCP packets to, for example, work around firewall problems.

RTSP is independent of both the underlying transport protocol and the protocol which is used to deliver the actual data. RTSP implements application-level reliability and can run over both reliable and unreliable transport protocols, including UDP and TCP. RTSP is also not tied to any transport-level connections and if the underlying protocol has connections the RTSP requests can be executed either within a single connection or multiple connections. The data delivery protocol is typically RTP, but other protocols such as RealNetworks' proprietary RDT can be used instead.

RTSP is similar to HTTP which has a number of advantages, including reusing the same basic structures, HTTP extension mechanisms, HTTP authentication mechanisms, and HTTP parsers for processing the protocol. It is also extensible, similarly to HTTP, with new methods and parameters.

RTSP defines a number of methods (similarly to HTTP GET, POST, PUT etc.), some of which are not necessarily supported by all implementations (similarly to HTTP PUT, DELETE etc.). These methods include the following:

- Playback and recording commands PLAY, PAUSE and RECORD. Playback and recording can be limited to a specified time range.
- Streaming setup and ending commands SETUP and TEARDOWN. The data delivery protocol is negotiated on setup.
- DESCRIBE and ANNOUNCE methods for delivering descriptions of media presentations, which can be described with formats such as SDP (Session Description Protocol) [139].

5.3.4. RTP

RTP (Real-Time Transport Protocol) [140] is a data transport protocol suitable for transmitting real-time data, such as audio and video. It supports multiple senders within a session for applications such as multi-participant multimedia conferences. Each media type is delivered as a separate session, which means that, for example, audio tracks for video presentations are delivered separately.

RTP can be run over both TCP and UDP (among others), and it also supports both unicast and multicast delivery. RTP does not provide any reliability guarantees by itself, but those may be provided by the underlying protocol. It does, however, provide a sequence number header field which can be used to detect out-of-order delivery and amount of lost packets. RTP also relies heavily on the underlying protocol for congestion control, which can also be profile dependent.

RTP specifies rate adaptation mechanisms which can be use to control the transmission rates more or less individually per receiver, even when multicast is used. These mechanisms include the following:

- Layered encodings can be used to construct the signal in several possible bandwidth and quality levels on the sources by transmitting the signal as hierarchical progressive layers, which are carried as separate RTP sessions. Receivers can then choose to receive only a subset of these layers. This obviously requires support for such encoding from the media format.
- Mixers and translators are RTP level relays which can be used for several purposes, including converting streams to reduced-quality versions or mixing together audio streams from multiple participants so that they can be transmitted as a single (low-bandwidth) stream through low-speed links. These relays can be configured for only some links or participants, so that the limitations do not affect those who have high-bandwidth links.

RTP is designed as a protocol framework which can be customized through modifications and additions to the headers as needed. The RTP specification in itself is not sufficient to specify the network level functioning of any particular application. For that additional profile and/or payload format specifications are needed. RFC 3551 [141], for example, specifies a profile for audio and video data, RFC 4629 [93] defines the payload format for H.263 video, and RFC 3984 [142] defines the payload format for H.264 video.

6. Streaming servers and media players

Online video streaming services typically resort to off-the-shelf streaming servers and media players for the implementation of the actual streaming and playback functionalities. Selections on these products can greatly affect the success of the service, as these products can have significant differences on many important issues, such as video quality and supported client platforms. The following sections describe some of the most commonly used software alternatives and their features from the viewpoint of video streaming service implementation.

6.1. Major streaming software providers

The streaming media software market, and especially the streaming video software market, is mostly shared between four major software providers. These providers and the primary media formats that they use in their products are Microsoft with Windows Media, Adobe with Flash, Apple with QuickTime, and RealNetworks with RealMedia. All of them provide both streaming servers and media players, and typically the choice of the media format largely dictates the use of the streaming servers and media players from the same provider, as these are rarely interoperable.

There are a number of factors to consider in choosing the media formats and streaming software components, such as:

- System requirements and availability of the media player.
- System requirements of the streaming server.
- Video quality.
- Streaming quality.
- Costs, which include licensing costs for the products and possibly also for the codecs that are used.
- Features and customization options for the user interface of the media player.
- Support for some desired special features of the server or the player. These may include, for example, support for live streaming, content protection, and integration to other components.

It is of course possible to use many alternatives simultaneously. On the client side the choice of formats may be given to the user or selected automatically depending on the user's environment. On the server side this may require multiple streaming servers or servers which support multiple formats. The use of multiple formats is, however, likely to cause additional implementation work and higher combined licensing costs.

6.2. Streaming servers

This section provides an overview of the streaming servers that are being offered by the four major players in the field. Two of them provide additional choices, as Adobe's product portfolio includes Flash Media Streaming Server and an enhanced functionality Flash Media Interactive Server, and Apple's offerings include QuickTime Streaming Server and its open source version, Darwin Streaming Server.

Table 2 lists these streaming server choices as well as their operating system requirements and pricing information for licenses that do not limit the number of simultaneous connections to the servers. The pricing information is included only as a rough indication of how the server costs may affect the choices on streaming servers. The following subsections summarize the media format and streaming protocol support of these servers and describe them individually in more detail.

Streaming server	Operating systems	Price for unlimited connections
Windows Media Services 2008 [143]	Windows Server 2008.	Included in OS
RealNetworks Helix Server 11 Unlimited [144]	Linux, Windows 2003, Solaris 8-10. [134]	No public list prices available
QuickTime Streaming Server 6 [145]	OS X 10.2-	Included in OS
Darwin Streaming Server 5.5.5 [146]	OS X, Windows, Linux, Solaris, and others.	Free
Adobe Flash Media Streaming Server 3 / Interactive Server 3 [147]	Linux, Windows 2003.	$995 / $4500 (+taxes)

Table 2: Operating system requirements and pricing of major streaming servers.

6.2.1. Format support

Table 3 summarizes the container format support of the streaming servers. It shows that support for the QuickTime based formats has become quite common, but otherwise there's not much overlap on the format support. It should be noted, however, that support for the same container formats certainly doesn't suffice to make those servers interoperable in any way, as there are differences in the supported codecs within those container formats as well as on the streaming protocols. H.264 encoded content, however, is supported by all of the servers that support the QuickTime based formats, but even this does not make those servers interoperable as the interoperability is limited by the supported protocols as is described in the following subsection.

Streaming server	Windows Media	RealMedia	Flash (FLV)	QuickTime	MPEG-4	3GPP
Windows Media Services [143]	X	-	-	-	-	-
Helix Server Unlimited version [144]	X	X	-	X	X	(1)
QuickTime / Darwin Streaming Server [148]	-	-	-	X	X	X
Adobe Flash Media Streaming Server / Interactive Server [147]	-	-	X	(2)	(2)	(2)

Table 3: Streaming server container format support. (1)=Requires Mobile Extension. (2)=Limited support, only H.264 content.

6.2.2. Protocol support

Table 4 summarizes the protocol support of the streaming servers. It is noteworthy that RTSP+RTP is supported by all servers except Adobe's servers. All servers provide support for HTTP tunneling.

Streaming server	Protocols
Windows Media Services	RTSP+RTP, HTTP (tunneling) [143]
RealNetworks Helix Server	RTSP+RTP, RTSP+RDT, MMS, HTTP (tunneling) [134]
QuickTime / Darwin Streaming Server	RTSP+RTP, HTTP (tunneling) [148]
Adobe Flash Media Streaming Server / Interactive Server	RTMP and variants (including those with HTTP tunneling) [103]

Table 4: Protocol support of major streaming servers

6.2.3. Windows Media Services

Windows Media Services (WMS) [143] is a streaming server which is an optional supplement for the Windows Server operating systems. There is some variance in the supported features based on the operating system edition. WMS can be used to stream live or on-demand Windows Media content for clients such as Windows Media Player and Silverlight and it supports both unicast and multicast streaming.

WMS supports RTSP streaming with optional HTTP tunneling. Support for the MMS streaming protocol was removed in WMS 2008 but the use of the MMS URL moniker (mms://) is still recommended as it provides automatic protocol rollover for the widest set of players (automatically selects either MMS(T/U), RTSP(T/U) or HTTP). Protocol rollover can, however, cause significant startup latencies if multiple protocols need be tried. Measurements by Guo *et al.* [149] showed that 22% of Windows Media streaming sessions had a rollover time longer than 5 seconds.

WMS supports a number of techniques for adapting to varying network conditions. These techniques are collectively known as intelligent streaming, and they include MBR

encoding with automatic bit rate version selection, stream thinning by reducing the frame rate and dropping video altogether for audio only delivery as a last resort. [150]

Fast Streaming is a group of techniques for improving the streaming quality [151]. These include Fast Start and Fast Cache among others. The aim of Fast Start is to reduce startup latencies by initially buffering data at higher than normal data rates, while Fast Cache is similarly used to fill the playback buffer with higher data rates during streaming to make the connection more tolerant to bandwidth fluctuations. There are, however, a number of downsides for using Fast Cache. First of all, the use of Fast Cache disables the use of Intelligent Streaming [150]. Secondly it works only with TCP based streaming. Thirdly it causes resource over-utilization including network bandwidth and CPU utilization. [149]

6.2.4. Helix Server

RealNetworks Helix Server [144] is a rare instance of a truly multi-format streaming server. It supports a number of container formats, codecs and streaming protocols and is available on Windows, Linux and Solaris platforms. It supports both on-demand and live streaming. Helix Server is sold in three varieties of which two have limits on the number of simultaneous streams (25 and 100) and one is unlimited.

Helix supports MBR encoding (known as SureStream) with automatic rate adaptation (Helix rate adaptation). It also has features to reduce startup latencies for on-demand and live content. These techniques are known as PlayNow and TrueLive, respectively. Helix Server Unlimited supports a number of features that are not available on the limited versions. These include support for multicast, content caching servers (Helix server functioning as a cache server in front of another), and redundant configurations for avoiding single points of failure within streaming and encoder servers. Cache/proxy functionalities can also be implemented with a separate Helix Proxy product. [134]

6.2.5. QuickTime Streaming Server and Darwin Streaming Server

QuickTime Streaming Server (QTSS) [145] and Darwin Streaming Server (DSS) [146] are streaming servers which are based on the same codebase. The former is a commercial product, which is available as part of Mac OS X Server and the latter is an open source version, which is available on multiple platforms. Both can be used to stream QuickTime, MPEG-4, and 3GPP content, both live and on-demand. Streaming is done over RTSP+RTP which can be tunneled over HTTP. Multicast is also supported. [148]

Streaming quality improvement techniques include instant-on streaming for reduced startup latencies and skip protection which is similar to Windows Media Fast Cache and uses aggressive buffering to reduce the effects of network bandwidth fluctuations. [152]

6.2.6. Adobe Flash Media Streaming Server / Interactive Server

Adobe Flash Media Streaming Server and Flash Media Interactive Server [147] are both streaming servers for Flash. The latter one has a number of additional features including support for multi-way applications, origin/edge server scalability, server side playlists, and a number of other features. Support for multi-way applications means that the Interactive Server is not just a streaming server but can also be used to implement interactive applications. It also supports server-side ActionScript code and plug-ins written in C++ for access control and other functionalities. Both servers support both on-demand and live streaming, and all codecs that are supported by Flash video.

Flash Media streaming uses the proprietary RTMP protocol (Real-Time Messaging Protocol) and its variants [103]. These variants include support for encryption and tunneling over HTTP.

Flash video doesn't support MBR encoding but the Flash servers do provide bandwidth detection functionalities, which can be used to simulate MBR by selecting the most suitable bit rate version of alternative separate video files. This does not however provide any kind of adaptation to varying network conditions. [103]

The use of proprietary streaming protocols means that Flash streaming cannot be performed by the other streaming servers considered here even though some of them support the streaming of H.264 video files, which are also supported by the latest Flash Players. There are however at least a couple of alternative streaming server implementations for Flash streaming, namely the commercial Wowza Media Server Pro [153] and the open source Red 5 flash streaming server [154], which is currently in beta stage. Both are Java 5.0 based and as such available for a number of platforms, including Windows, Linux, and OS X.

6.3. Media players

This section provides an overview of those media players that are being offered by the four major players in the field. Of these, Microsoft provides two distinct media player products. All of the players that are considered here support the playback of streaming video and embedding of the player into a web page. There are of course other good media players such as VLC Player [155] and MPlayer [156], which are free and cross-platform, support a wide variety of formats, and there are even browser plug-ins available for some browsers. They are not, however, as widely used as those players that are considered here and they have some significant limitations when it comes to streaming and embedding.

Table 5 lists the media player choices as well as their operating system requirements and basic information about their installation base. Media player availability and deployment figures can be quite important for online video services which aim to be available to as many users as possible. First and foremost, players should be available for

installation for at least the most common operating systems, but it is also a big advantage if the player is likely to be already installed on most users, so that they can immediately use the service without additional software installations.

Media player	Operating systems	Significant pre-installations	Penetration (1)
Windows Media Player 11 [157]	Latest version: Windows XP and Vista Older versions: other Windows versions, Mac OS X, Mac OS 8.1	Pre-installed in almost all Windows operating systems.	82.2%
RealNetworks RealPlayer 11 [158]	Latest version: Windows XP and Vista Older versions: Mac OS X, Linux	Pre-installed on many Linux distributions. [159]	47.1%
Apple QuickTime Player 7 [160]	Mac OS X 10.3.9-, Windows XP and Vista	Pre-installed in Mac OS X operating systems.	66.8%
Adobe Flash Player 9 [161]	Windows 98 and later, Mac OS X 10.1-, Linux, Solaris 10	Pre-installed in many browsers and operating systems.	98.8%
Microsoft Silverlight 1.0 [162]	Windows XP and later, Mac OS X 10.4.8-. Linux support is in progress as open-source Moonlight project.	-	Unknown

Table 5: Operating system requirements and availability information of media players. (1)=Installation percentage (of all player versions) on Internet-enabled PCs in mature markets (6 selected countries) on March 2008. [163]

The following subsections summarize the media format and streaming protocol support of these players and describe them individually in more detail.

6.3.1. Format support

Table 6 summarizes the container format support of the media players. Like with the streaming servers, support for the QuickTime based formats has become quite common, but otherwise there's less overlap on the format support.

Streaming server	Windows Media	RealMedia	Flash (FLV)	QuickTime	MPEG-4	3GPP
Windows Media Player [164]	X	-	-	1.0-2.0	-	-
RealNetworks RealPlayer [158]	*(1)*	X	*(1)*	*(1)*	*(1)*	*(1)*
Apple QuickTime Player [160]	-	-	-	X	X	X
Adobe Flash Player [161]	-	-	X	(2)	(2)	(2)
Microsoft Silverlight [162]	X					

Table 6: Media player container format support by default installation, additional formats may be installed separately for some players. (1)=Support varies by operating system and other installed players. (2)=Limited support, only H.264 content.

It should be noted that many media players can actually open formats which they do not natively support by opening them with the help of other installed players, which will be embedded inside them. RealPlayer, for instance, can play all of the container formats that are mentioned in Table 6 by opening them in an embedded Windows Media Player, QuickTime player or Flash Player, if these players are available.

6.3.2. Protocol support

Table 7 summarizes the protocol support of the media players. The protocol support has a similar pattern to that of the streaming servers as RTSP+RTP is supported by them all except Adobe's players and Silverlight. All of the players also support HTTP tunneling, which is actually the only alternative with Silverlight.

Media player	Streaming protocols
Windows Media Player 11	RTSP+RTP, MMS, HTTP (Tunneling) [143]
RealNetworks RealPlayer 11	RTSP+RTP, HTTP (Tunneling) [134]
Apple QuickTime Player 7	RTSP+RTP, HTTP (Tunneling), [165]
Adobe Flash Player 9 [161]	RTMP and variants, HTTP (Tunneling) [103]
Microsoft Silverlight 1.0 [162]	HTTP, HTTPS

Table 7: Media player streaming protocol support

6.3.3. Windows Media Player

Microsoft Windows Media Player (WMP) [157] is a media player which is pre-installed on almost all Windows systems.

The possibility for media player user interface customizations can be an important feature for building visually appealing video sites. One important customization is the possibility to replace media player buttons (play, stop, etc.) with customized button graphics. WMP has been problematic in this respect, as while customizations could have been done with the help of scripting, the scripting support has been available only when the player is embedded as an ActiveX control. In practice this has meant that scripting has worked directly only in Internet Explorer and users of Mozilla based browsers have been required to install a separate ActiveX plug-in for such support [166]. The situation changed somewhat recently as Microsoft released a new Firefox plug-in which supports scripting. This plug-in, however, is only supported by Windows XP SP2 and Windows Vista, and hence the problem still remains if the web site should work with older Windows versions [167].

6.3.4. Silverlight

Microsoft Silverlight [162] is a recent rich Internet application technology which can be seen as a direct competitor to Adobe Flash and other RIA technologies. Similarly to Flash, there is no generally used standalone Silverlight player as it is solely used as a browser plug-in. It is currently available for Windows XP and later and Mac OS X

10.4.8 and later with browser support for Internet Explorer, Safari and Mozilla-based browsers. Linux support is in development by a third party project Moonlight. The current stable release is 1.0 and version 2.0 is in beta stage.

Silverlight supports the playback of Windows media content with streaming or progressive downloading. DRM will be supported by Silverlight 2.0. Silverlight makes it is possible to build a custom video player with custom controls and look and feel, something which was not possible to do in a cross-browser way with the embedded Windows Media Player.

6.3.5. Flash Player

Adobe Flash Player [161] is a lightweight client runtime for implementing rich Internet applications. It has a wide operating system and browser support and it is ubiquitously installed on most Internet-enabled desktops, which makes it an attractive platform for application and service implementers.

There is a significant distinction between a media player such as Windows Media Player and a client runtime such as Flash Player. Flash Player is not primarily an audio or video player but a player for Flash applications. These applications utilize EcmaScript based ActionScript scripting, vector graphics, multimedia content, and so on. Flash Player can't be simply embedded on a web page and given a Flash video (FLV) URL. Instead, one must first make or acquire a video player Flash application, which is embedded on the web page, and this Flash application in turn can play FLV files, among other things.

Flash Player didn't even have true video support before version 6, which was released in 2002. Before that, video had to be simulated by showing still images in sequence [168]. Flash Player 6 added the support for embedded video (video contained within the main SWF file) and streaming video. Version 7 added support for progressive downloading of video files [115]. These player versions supported only two video codecs, the Screen Recording codec (for screen recordings) and the Sorenson Spark video codec, which is also known as Sorenson H.263.

Flash Player 8 added support for the On2 VP6 video coded, which provides better video quality and support for an alpha channel. Flash Player 9 Update 3, which is currently the latest version, introduced support for the H.264 video codec alongside limited support for MPEG-4, MOV, and 3GP container formats. [103]

6.3.6. RealPlayer

RealNetworks RealPlayer [158] is a multi-format media player which supports a number of video formats in addition to the RealNetworks' proprietary RealVideo. RealPlayer can play Windows Media, QuickTime and Flash videos but these require that the appropriate

additional players are also installed and RealPlayer actually plays them by opening those other players inside itself.

One particularly interesting feature of the latest RealPlayer version is that it is also a stream grabber that is capable of downloading and recording video streams that are encoded in a number of formats, including RealMedia, Windows Media, Flash, and QuickTime, except those that use DRM. It provides a browser plug-in which enables the user to download videos directly from web sites with a single click, even if RealPlayer itself is not running. This works with sites such as YouTube and even live streams can be recorded. [169]

RealPlayer has several possibilities for user interface customizations. It has scripting support for controlling the player with custom playback buttons, and the standard playback controls can also be embedded individually as separate components anywhere in the web page. RealPlayer also has extensive SMIL [170] support which includes interactive elements. [171]

6.3.7. QuickTime Player

Apple QuickTime Player [160] is a multi-format media player which can also play interactive content in the QuickTime format, and therefore it can be used also for RIA applications.

Interactivity and user interface customizations can be accomplished with QuickTime Sprites, which enable vector based animated graphics and are contained in a QuickTime file as a separate sprite track [172]. User interface customizations are also possible with scripting support and Flash tracks, but the supported Flash versions have been lagging behind Adobe releases, and as of QuickTime 7.1.3 Flash support has been disabled by default [173]. QuickTime also has SMIL support, but it doesn't support those SMIL constructs which would enable interactivity [174].

QuickTime also has a skinning feature known as Media Skins with which the stand-alone QuickTime player can be skinned to any shape and size with customized playback controls. Skinning is a common feature in media players but QuickTime Media Skins is different in that skins are defined inside media files, and hence content producers can decide how the player looks like while playing their content. [175]

7. Overview of the Media CMS

Media CMS is an extensible audio and video delivery platform, which is intended to become a full-blown content management system with a wider range of functionalities. It has already been proven to work in the real world, as it has been powering a couple of high profile online audio and video services for some time now.

The rest of this thesis concentrates on describing the implemented system. This chapter will give a high level overview of the system including historical background to the project that led to the creation of the system, the design principles that were followed during the project, the most fundamental features that it contains, and an architectural overview of the system and its layers. The following chapters will describe the most important components in more detail.

7.1. Introduction to the system

Media CMS intends to be a full blown content management system in the future, as the name implies. At the moment it still lacks some typical CMS functionalities, but it already provides a solid basis for building audio and video services. It is designed as a platform for building secure and scalable multimedia services ranging from video enhanced news sites or multimedia rich product presentation pages to heavy-duty video sharing sites or online music stores. Media CMS is the end result of a work which was extended from an implementation project which originally had a significantly narrower scope.

7.1.1. History

While Media CMS in its current form is a generic platform suitable for multiple uses, that wasn't the original intention. It has its roots on an implementation project aiming to produce the audio and video delivery mechanisms for a couple of well known high volume online audio and video services. It became evident in the early stages of the project that there were a lot of anticipated future needs and technological choices which were not yet fully known or defined but nevertheless needed to be taken into account on the design, at least on some level. These included specifics of data models, user interface features, media formats, and integrations to various off-the-shelf components and back-end systems, such as streaming servers, content sources, and user interface implementations. Hence it made sense to aim for architectural decisions which would be generic and flexible enough to accommodate to these anticipated needs without major modifications.

The project had extremely tight schedules, and lack of time was seen as the biggest challenge from the beginning. Hence there was an obvious need to save time where-ever it was possible without making too many sacrifices on the necessary features or future extensibility. It was also clear that it was not possible to determine all the necessary

features right from the beginning, and hence the system should be implemented so that future additions can be combined later to the existing structures as seamlessly as possible.

XML-based file formats and interfaces were an obvious choice for providing some level of extensibility. Use of standard formats was also seen as a clear benefit from a number of viewpoints, one of which was the possibility to conserve some time. The reasoning behind this was that mature standards tend to be well thought and tested solutions which should lead to solid solutions without taking the time to design new formats. And while it was evident that there wasn't any existing single standard solution which would provide everything that was required in one package, it would still make sense to combine and extend such solutions as far as possible to minimize the burden of proprietary designs. Use of standards could also provide other benefits, such as benefits in integrations to systems that utilize the same standards, flattening the learning curve for anyone familiar with those standards, and providing existing documentation.

It was also clear that, given the time constraints, there wouldn't be time to fully implement any standards for the standards sake, and hence the intention was to implement what was currently needed and to add new features as needs arise. This should also provide clear development paths towards the directions that were defined by the standards.

Some technological choices were already made, for various reasons, before the project properly began. The main implementation language was selected to be Java. The database product that would be used was selected to be MySQL [176]. The initial media format was selected to be Windows Media, although it was evident that support for additional formats could be needed at a later stage. The selection of the media format also led to the selection of Windows Media Services as the streaming server and Windows Media Player as the embeddable player implementation, as it was clear that off-the-shelf components would be used for such purposes. Microsoft Silverlight, which has a number of advantages on the client-side implementation as compared to Windows Media Player, was not yet available at that time.

7.1.2. Intended usage and features

Media CMS was designed to be the basis for high profile online video and audio services which were known to have quite a lot of users and content. It was clear that the system should be both secure and scalable. Reliability and failure-tolerance were also important issues, and every effort was made to design the system so that all components could be clustered and there wouldn't be any single points of failure.

The first services that were implemented were business-to-consumer (B2C) type and featured exclusively professionally generated content. These kinds of services were

therefore the main focus in the development work. However, the possible application to services utilizing user generated content, such as video sharing sites, was kept in mind.

The initial content was known to be streamable on-demand content, but it was also known that downloadable content and live streaming would be needed at a later stage, and all of these options should be provided. It was evident that, given the resource constraints, a number of low-level details, such as delivery mechanisms and media player implementations, would be implemented with off-the-shelf components, such as those described in chapter 6. Therefore in this regard, design efforts were concentrated on providing compatibility and extension points for as many of such components as possible. This also included compatibility requirements for the various video formats and networks protocols that can be utilized by such components.

It was also clear from the start that the data contents of the system would not be simply individual audio and video files but structured and interconnected presentations consisting of several media files and metadata properties. This became one of the defining features throughout the design of the system, and it also steered the development towards the direction of a multi-purpose content management system.

7.1.3. Design principles

A number of design principles were kept in mind throughout the design of the system. These included the following:

- **Performance**. The system is designed for heavy duty services and hence each component should be adequately optimized, and the system in whole should be scalable and free from obvious and hard to fix bottlenecks.
- **Reliability**. Popular and highly visible services cannot tolerate downtime. The system should recover gracefully from common failures and it should allow clustered deployments without single points of failure.
- **Security**. A system is only as secure as its weakest link and hence every effort should be made to avoid such weak links. Security is also a combination of different factors, including human errors, and hence the system should be designed so that such errors are not too easy to make. Even some redundancy is not a bad thing when it comes to security checks, as if one of them happens to fail, the others may still prevent an attack, or at least minimize the consequences.
- **Extensibility**. The system is designed as the basis for a variety of services which may have different needs, and not all of them can be known or implemented straight away. It is essential that these features can be added later as needed, and they should integrate as seamlessly as possible with the existing features.
- **Separation of concerns**. The system should consist of components which have clearly specified roles. The individual components, in turn, are typically

composed of subcomponents which should similarly have clearly specified roles and concerns.

- **Maximal code reuse**. The implementation work should aim for well thought and reasonably multi-functional components which can be reused within different parts of the system. This should save some time both in the initial implementation and during modifications and additions.
- **Internationalization**. The system should be designed for global audiences from the start. This has to be taken into account in a number of places. All textual content, for example, should be processed as Unicode [177] and it should be possible to define all texts in multiple languages.
- **Aim for generic solutions, but do not lose functionality**. It is good to have solutions that are flexible enough to be suitable in a number of situations, but it is also easy to overdo it. A solution that is too generic may be hard to use and require its user to tediously fill in the details that are left out to keep the solution generic. This problem can be tackled with layered designs where the lower layer implements the generic parts and the upper layer contains implementations which utilize the generic layer and are likely to be suitable for the majority of use cases. There are also a number of well known design patterns which can be used for the same purpose, such as the Visitor and Strategy patterns [178]. The implementation of the Media CMS utilizes a number of such patterns in various places.

7.2. Architectural overview

On the architectural level, Media CMS consists of a number of individually deployable components which can be conceptually grouped into different layers, although depending on the configuration of the system, these layers may not be strictly stacked on top of each other. Each of these components has its specific purposes and responsibilities and well-defined interfaces and connections to other components. Figure 1 contains an overview of the conceptual layers of the system and the most significant components that are contained within them. These will be described in more detail in the following sections.

It should be noted that the components and the interconnects between them have been designed so that there are as few limitations as possible for the distribution of the components within a set of server nodes. All of the components that are pictured in Figure 1 can be clustered or replicated on any number of nodes and there are no architectural restrictions on which components should be located on which nodes. It is therefore possible to run the entire system on a single server node or to distribute each and every component on separate, and possibly multiple, nodes.

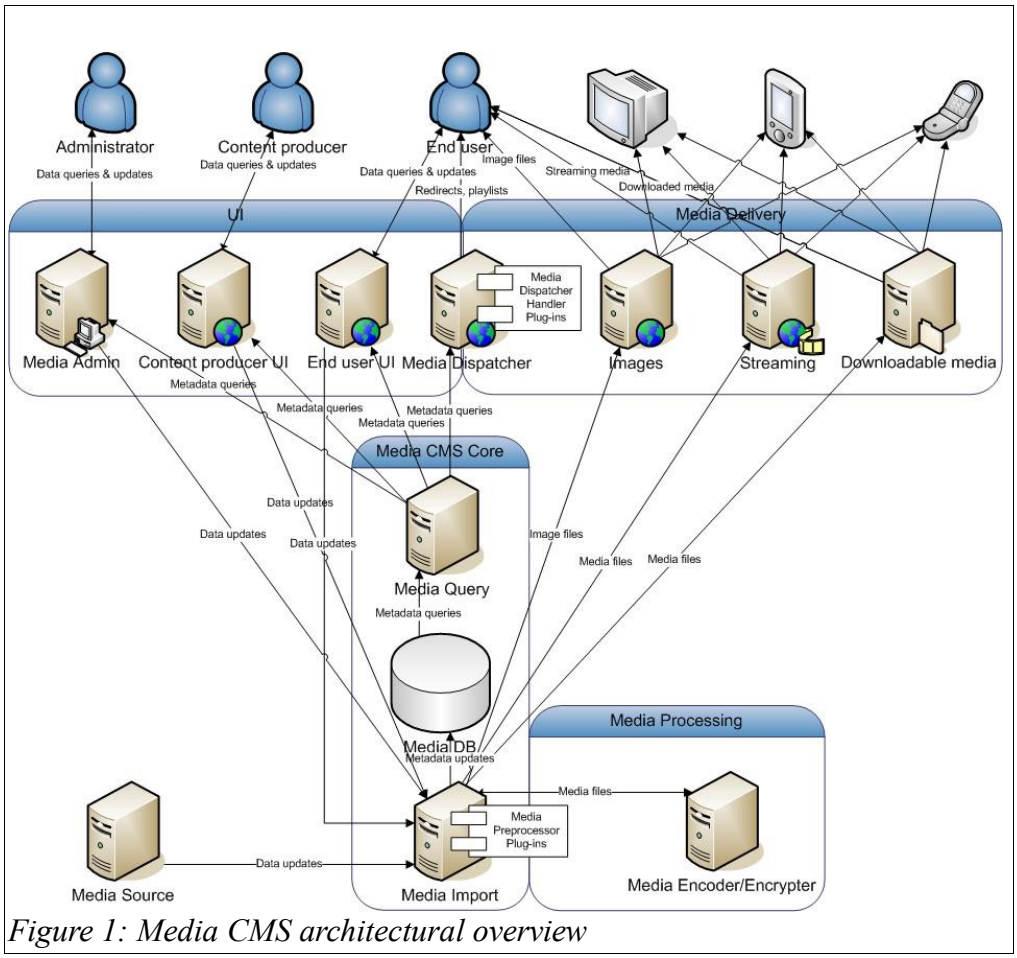

Figure 1: Media CMS architectural overview

7.3. The Core layer

The heart of the system is the Media CMS core. It consists of a database and the components that implement the external interfaces that provide access to the database. The core is completely data model, media format, delivery method, and user interface technology agnostic. Processes that are specific to such structures or technologies are handled either by plug-in components or components completely outside the core.

7.3.1. The Media Database

The data contents of the system consist of the actual media data files combined with presentation data. The presentation data consists of presentation structure information, which can be used to link and annotate individual files in a number of ways, and of various descriptive metadata properties, such as titles and description texts. All the presentation data is stored within the database at the centre of the core. This database is described in more detail in chapter 8.8. The actual media files are stored as files outside the core in one or more configurable locations.

7.3.2. Core XML Interfaces

Media Query and Media Update are the components that provide XML interfaces for querying and updating the data contents of the system. Basically the core is like an XML database that is implemented internally on top of a relational database.

The design of the core decouples the other components from the internal data structures of the core database, which are bound to change over time. The XML formats are expected to stay relatively constant and backwards compatible despite the possible internal database changes. This design also provides a layer of security and data integrity, as access to the database is allowed only through the XML interfaces.

7.3.3. The update-storage-query architecture

The core can be thought of as an update-storage-query pipeline where all the updates are done at one end (Media Update), and all the queries are done at the other (Media Query). This design provides some important benefits over a combined query and update interface in regard to scalability and security.

The core components can be arranged in a couple of significantly different ways on the network level, depending on who needs to have access to the update functionalities. These arrangements are show in Figure 2.

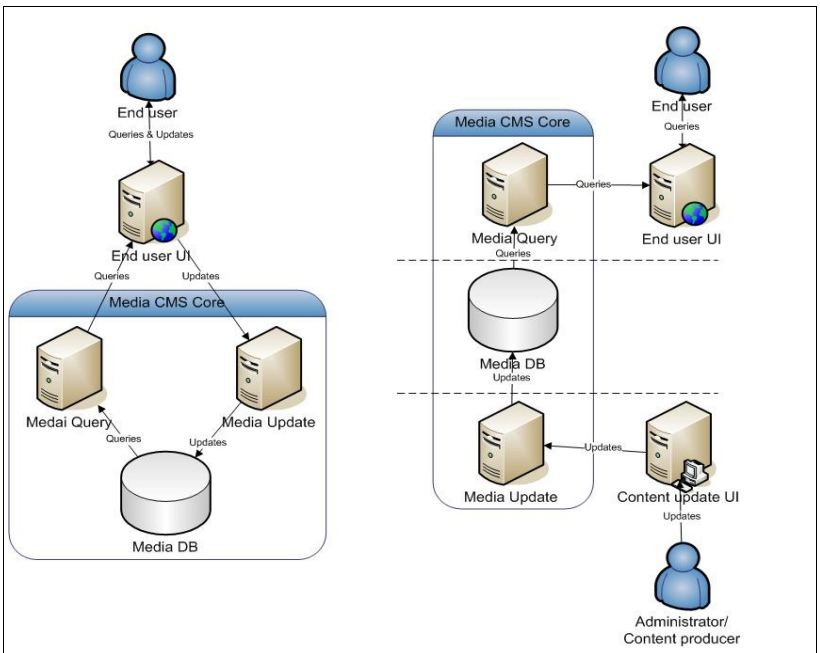

Figure 2: Core component network level arrangement options

The left hand side of Figure 2 shows a typical arrangement in a service which utilizes user generated content. In this case, both the update and query interfaces need to be available for the end-user interface, and the benefits of separating these interfaces may not be immediately obvious. The right hand side of Figure 2 shows a typical arrangement in a service which contains only professionally generated content that is not updated through the regular end-user interface. In this case, access to the update interface can be restricted already on the network level with firewalls or IP access lists. This can significantly improve security. The database can be also configured as a database cluster, in which updates are done in one read-write database on the secured LAN side and those

updates are replicated to one or more read-only databases on the query side, which may be located outside the inner firewall.

It should also be noted that even in the left hand side case the separation of the interfaces can have a number of advantages, as the interfaces can be individually clustered and distributed on the server nodes, and it is also possible to perform maintenance breaks on the update side without harming the query side.

7.4. The Media Processing layer

The data contents of the system can originate from a wide variety of content sources, including content that is uploaded from the various user interfaces and content that is automatically transferred from integrated back-end systems, such as radio and TV broadcasting systems. Some of these sources can provide content formats that can be directly distributed to the end-users, while others may first require some transformations. These transformations can include, for example, encoding the content to a different format or encrypting the content with DRM mechanisms. These transformations are handled by the Media Processing layer.

The Media Processing layer is implemented by Media Pre-processor plug-in components that are attached to the Media Update interface. These plug-ins can be executed selectively based on the presentation data contents that are sent to the update interface alongside the media files. Some of these plug-ins may utilize external components for performing the actual transformations, but, from the viewpoint of the Media Update interface, these are implementation details that are hidden behind the plug-in interfaces. In addition to transformations, the plug-in components can also perform various data gathering operations, such as extracting the technical details about the tracks of a media file and recording this data within the presentation data, which will be stored to the media database. After the transformations have been performed, the Media Update interface will transfer the resulting files to configurable locations from which they will be accessible to the delivery servers.

7.5. The Media Delivery layer

The Media Delivery layer is responsible for delivering the actual media files to the end users' browsers, FTP-clients, mobiles and so on. These files are typically requested through some user interface implementation, such as by a media player that is embedded on a web page. They may be also requested by, for example, portable players through URLs that are contained in podcast RSS feeds [179]. Media files can be requested by their direct URLs from various delivery servers, but more typically they are routed through the Media Dispatcher, which is described in subsection 7.5.2.

7.5.1. Delivery servers

The Media Delivery layer can contain a diverse set of distribution servers. These may include streaming servers for different file formats and protocols, HTTP and FTP downloading servers, image servers, and origin servers for hybrid P2P delivery services, such as those used by Joost [180].

7.5.2. Media Dispatcher

Media Dispatcher is a component that in some ways connects the UI layer to the Media Delivery layer, and hence it can be seen as part of both layers. Basically it can provide a number of functionalities and output formats with simple requests which specify the id number of a media presentation and possibly some additional controlling parameters. Media Dispatcher processes most of the requests by performing queries to the Query Interface. It can be used for a multitude of purposes, which include the following:

- **Decoupling of request URLs and storage locations**. URLs that are written to, for instance, HTML pages can always point to the same address (that of the dispatcher) despite the fact that the actual content may be distributed to several locations and also moved between locations without modifications to the UI.
- **Content selection**. Media Dispatcher can select and return the best content format alternative based on the parameters of the request.
- **Content presentation**. Media presentations can be formatted differently for different clients, for example as SMIL presentations for RealPlayers or as ASX playlists [181] for Windows Media Players.
- **Fault tolerance**. Generated playlists can contain alternative locations for the content as fallback URLs for the primary one.
- **Load balancing**. The Media Dispatcher can direct individual requests evenly to alternative delivery servers for distributing the server load.
- **Security checks**. The Media Dispatcher can implement authorization and security checks on behalf of the delivery servers. Authorization information can be delivered securely to the delivery servers with additional security check parameters that are protected by cryptographic hashing functions.

Most of the functionalities of the Media Dispatcher are actually performed by configurable Media Dispatcher Handler plug-ins which are selected based on the parameters of the requests. These plug-ins can be dependant on individual media formats or even on service specific details, but the Media Dispatcher itself is fully generic in nature.

7.6. The UI layer

The UI layer contains the user interface implementations for updating and viewing the contents of the Media CMS. These may include a number of separate user interfaces for different user groups, including content producers, administrators, and actual end-users.

Media CMS provides some out of the box user interfaces and user interface components, but typically these interfaces are custom built for each individual service, in which case they can be seen to either form part of an extended UI layer or be completely separate of the Media CMS. The customized user interfaces can be implemented with any user interface technologies that can be integrated to the Media Query and Media Update XML interfaces.

8. The Media CMS data model and database structures

Media CMS is designed as a platform which can support a wide range of diverse services and service types, which may have significantly different needs for data models, media formats, delivery methods, and integrations to back-end systems and third party components. This diversity poses a number of challenges for the design. This chapter describes how those challenges are tackled on the data model level with flexible and extensible data structures and presentation formats.

8.1. Requirements for the data model

Media CMS was originally designed to function as the basis for a couple of online audio and video services which already featured a number of requirements for the data model. The identified requirements included the following:

- The data model should be able to express video, audio, and image files and their technical details, such as bit rates and resolutions. These should be processed in a generic way, so that system is compatible with a wide variety of file formats and codecs, such as those that were described in sections 4.2 and 4.3.
- Furthermore, it should be possible to store the technical details of media files on the level of individual tracks or track groups for container formats like those that were described in section 4.3.
- It should be able to express alternative file versions and presentation formats from which the best one can be selected based on the users' preferences or the automatically detected capabilities of the users' browsers or terminals. These file versions can be targeted, for example, for different media players, delivery methods, usage purposes, network protocol alternatives, network speeds, and users' language preferences. These alternative versions should be linked together so that they can be easily managed as a single item.
- The media files can be delivered from multiple delivery locations, which can utilize different delivery methods, network protocols and off-the-shelf server components, such as the streaming servers that were described in section 6.2.
- It should be possible to group and categorize media files and media file structures. Furthermore, these groups can be nested in hierarchies, and a single object can belong to several groups.
- It should be possible to describe and annotate the various individual objects and the structures that they form with, for example, textual properties like titles and description texts.
- It should be able to specify further interrelations between the various objects and object structures beyond alternative file versions or groupings. These can include,

for instance, the interrelations between a full-length video and a screenshot or trailer video that is extracted from it.

The specifics for some of these requirements were rather vaguely defined at the beginning and it was clear that the data model should be flexible enough to accommodate to changes in the service specific details. There were also some preliminary plans to support user generated content and various web 2.0 features, such as commenting and rating of the content. It was determined that, while these features were not in the short term implementation plans, the data model should be extensible with such features later.

8.2. The foundation of the data model

The list of data model requirements was a long one and posed some serious challenges. The data model should not only include all the necessary features, but it should also implement them in such a way that it lends itself to a scalable high-performance implementation which can be done within strict time limits.

The design phase began with a survey of the available markup languages and file formats that could provide at least some of the necessary features and could therefore function as the basis for the data model design. The best alternative that was found was the Synchronized Multimedia Integration Language (SMIL) [170], which provided at least some support for many of the required data model features. It was also an extensible XML based format and a widely used standard, which was directly supported by a number of media players, mobile phones, and other clients.

The RDF/XML syntax [182] of the Resource Description Framework (RDF) [183], which is also used by SMIL, was additionally chosen as the format for the metadata properties of the various objects. However, it was determined that the RDF properties should be defined within the individual objects that they describe, as the use of a separate header section, such as that used by SMIL, was not appropriate in the context of the entire data model.

Together SMIL and RDF provided a good foundation for the data model, but there were still a lot of requirements to be fulfilled. It was decided that these would be handled by proprietary extensions.

8.3. SMIL-based structures

SMIL is an XML based language for expressing multimedia presentations. It has structures for, for example, expressing the locations of the actual media files, presentation layouts, animation and interactive elements. Most of these are not needed within the Media CMS data model and hence the data model contains only a subset of the SMIL feature set. Currently this subset contains the features that are required for SMIL 2.1 integration set conformance, with a few additions.

The Media CMS data model has a number of physical representations, including the database presentation, the data model Java classes, and the Media XML format. Of these, the Media XML format is the most commonly used and will also be used in the data model examples of this chapter.

Figure 3 is an example of a simple Media CMS data model media presentation, which utilizes SMIL-based structures with a few Media CMS specific additions. The innermost video elements represent individual video files and specify their container formats and file locations. These elements are contained within a switch element, which is used to select the first acceptable nested element based on the system test attributes that are defined for the nested elements. In this example the first video element contains a `systemBitrate` attribute which specifies that the user's bandwidth must be at least 50000 bits per second for that file version, otherwise the switch will select the second nested element, which does not contain any system test rules. The switch element is contained in a video element, which is the root element of the example media presentation, and this element also specifies a descriptive title for the presentation. The title is specified with the title element from the Dublin Core Metadata Initiative [184], although the XML namespace declarations are omitted from this example. The use of video elements as hierarchy roots is a Media CMS specific extension, as is the definition of RDF properties directly within such elements. Otherwise constructs in this example are standard SMIL features.

```
<video>
     <rdf:Description>
          <dc:title>Video presentation example</dc:title>
     </rdf:Description>
     <switch>
          <video src=”file:/hi.qt” type=”video/quicktime”
                 systemBitrate=”500000” />
          <video src=”file:/lo.wmv” type=”video/x-ms-wmv” />
     </switch>
</video>
```

Figure 3: SMIL-based media presentation structures

8.4. Expressing the technical details of media files

Media CMS stores the data contents of the system in two places: the actual media data files are stored in configurable disc locations and everything else is stored into the database. Sometimes it is necessary to know the technical details of the data files without the examination of the actual files, and therefore these details can be stored also into the database. However, SMIL does not contain structures for expressing such details and therefore those are described with proprietary extensions.

Figure 4 contains an example of specifying the technical details for a video file in the ASF (.WMV) container format, which contains a couple of alternative video tracks

within a track group and a single audio track. All of them have specified bit rates but the rest of the technical details have been omitted in this example. What is noteworthy is the reuse of the switch element and the system test attributes for expressing the exact usage rules of the contained tracks. In this case, the video file is an MBR file with two alternative bit rate versions of the video content.

```
<video type="video/x-ms-wmv">
     <trackGroup>
          <switch>
               <videoTrack systemBitrate="512000">
                    <bitrate>512000</bitrate>
                    ....
               </videoTrack>
               <videoTrack>
                    <bitrate>256000</bitrate>
                    ....
               </videoTrack>
          </switch>
     </trackGroup>
     <audioTrack>
               <bitrate>64000</bitrate>
               ....
     </audioTrack>
</video>
```

Figure 4: Expressing the technical details of media files

8.5. Groups

Another feature that is missing from the SMIL language is the ability to express arbitrary groupings of media presentations. Therefore these are also implemented with proprietary extensions.

Figure 5 contains an XML fragment which specifies some nested groups and contained video elements. This example also demonstrates how the individual objects can have specified types and identification properties.

```
<group mediaType="category" id="10" name="news">
     <group mediaType="subcategory" id="100" name="sport">
          <video mediaType="video" id="1234" name="video1" />
          <video mediaType="video" id="5678" name="video2" />
     </group>
     <group mediaType="subcategory" id="200" name="politics">
</group>
```

Figure 5: Grouping elements

The objects of the Media CMS data model are known as media objects and each of them has a specified type and identification properties. In the Media XML format the type can be defined with the `mediaType` attribute, and in case it is omitted an element specific default value is used, such as "video" for video elements. All media objects also have unique id identifiers, which can be automatically generated for new objects. The objects can also have more descriptive name identifiers.

The example of Figure 5 defines a top-level category object which contains a couple of subcategory objects, one of which contains a couple of video elements.

8.6. Relations

One of the listed requirements for the data model specified that it should be possible to include a single media object in several object groups (which may not be nested). In practice this means that the data model of the Media CMS cannot consist of simple tree structures, and therefore the data model is actually a directed acyclic graph (DAG), which contains all of the objects within the database. This also means that such structures cannot be expressed in the Media XML with simply nested elements without at least duplicating some elements. The Media XML format actually provides a number of possible arrangements for these structures.

Figure 6 shows a couple of the possibilities for connecting video elements to a couple of category groups. It also introduces the concept of relations. The DAG structure of the Media CMS data model basically consists of media objects and relations that connect them. Both media objects and relations can be explicitly expressed in the Media XML format as is shown in the example.

```
<group mediaType="category" id="1" />
<group mediaType="category" id="2" />

<video id="3">
<relation srcId="1" destId="3" />
<relation srcId="2" destId="3" />

<video id="4">
     <relation axis="parent" destId="1" />
     <relation axis="parent" destId="2" />
</video>
```

Figure 6: Expressing the relations between objects

The example first defines a couple of category objects with unique ids 1 and 2. Then it defines a video object with id 3 and separately connects that object to the categories to which it belongs with explicit relation elements. These relation elements refer to the source and destination objects with their unique ids, but name references are also possible. The relations are by default parent-child relations and hence the categories are specified as the sources (parents). The category relations for the second video element with id 4 are defined differently by specifying the relation elements inside the video element. In this case the relation sources are not explicitly specified, in which case the source is assumed to be the containing object (id=4). Likewise, if the destination is not specified it is by default the element that the relation element contains. The relations for the second video element are also reversed with an explicit axis specification so that the relations become child-parent relations (the axis name refers to the role of the destination) and hence the categories can be defined as destinations (parents).

It is also noteworthy that in the Media XML format the standard SMIL syntax of nested elements is actually simply a shortcut for a relation with default values. Hence the two elements structures of Figure 7 yield the exact same results. Figure 7 also shows that relations have also types and the default type is “component”. Other relation types can be used for, for example, specifying that a video object is a “trailer” of another.

```
<switch id=”1”>
     <relation axis=”child” type=”component”>
          <video id=”2”>
     </relation>
</switch>

<switch id=”1”>
     <video id=”2”>
</switch>
```

Figure 7: Relation shortcut syntax with default values

8.7. Media type and relation type hierarchies

Both the media types and the relation types form inheritance hierarchies, which can be extended with new types as needed. The standard hierarchies are shown in Figure 8.

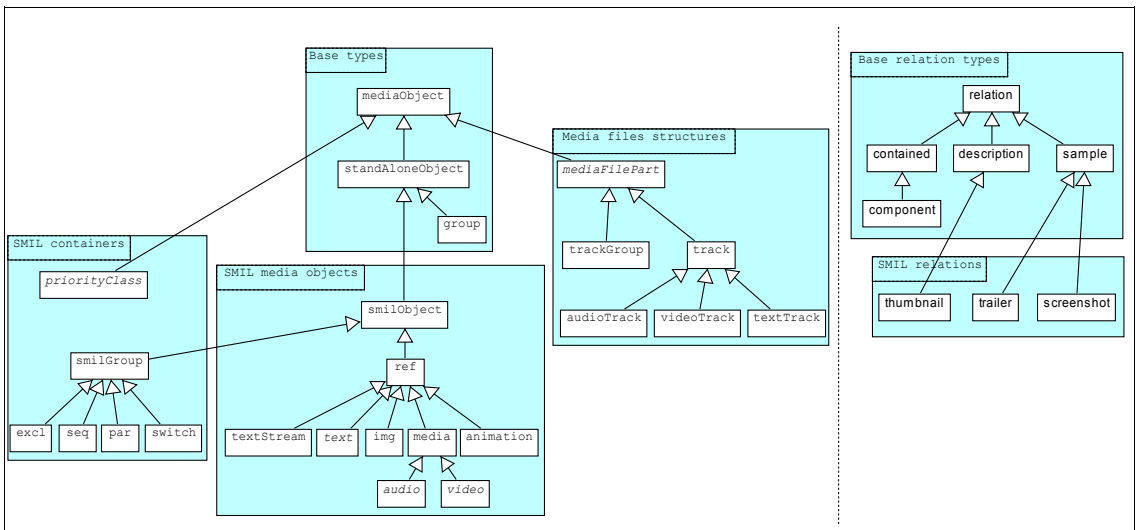

Figure 8: Hierarchy of the standard media types (left) and relation types (right)

The type hierarchies are significant in many respects, as inherited types are also instances of their parent types, and hence objects can be, for instance, queried through the Media Query interface by their parent types. Types are also an important aspect of the self-documenting nature of the data model, as descriptive types indicate the usage purposes of the objects, and the inheritance hierarchy further describes their properties and relations to other types.

8.8. Database implementation

The Media DB database implementation of the data model storage tables is a relatively straightforward relational representation of the Media CMS data model. The most significant tables of this representation are shown in Figure 9.

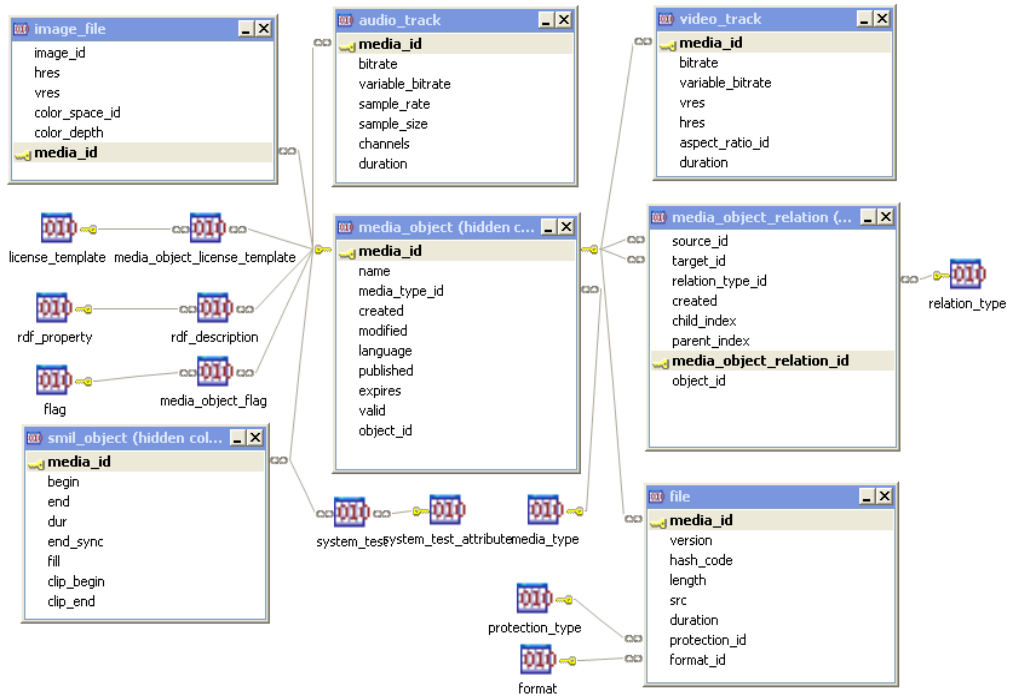

Figure 9: Media DB tables for the storage of media object and relation data.

The media_object table maps to the type hierarchy root type. It contains one row for each media object. The media type inheritance hierarchy is implemented as media type specific tables, which are joined to the main table by foreign key references. The database, however, contains significantly fewer tables than there are individual media types, as most of the types do not define any additional data fields that would need new storage tables. There are some differences in the naming of the tables and the media types, as the img media type maps to the image_file table and the ref media type to the file table. Relations between media objects are contained in the media_object_relation table. This table contains only direct parent-child relations as the reversed relations are already normalized in the Media Update interface as parent-child relations.

8.9. Data updates

All updates to the database are performed in the Media XML format through the Media Update interface, which ensures that the data contents remain in a consistent state and adhere to certain data validity constraints. This interface supports an additional `updateType` attribute, which can be defined individually for each media object and relation. This attribute specifies whether the media object or relation will be added, updated, overwritten, or removed.

The update interface is also responsible for copying the source media files that are referenced by the Media XML files to their correct delivery locations and marking those locations to the database. This process may also involve transformations to both the actual media files and on the presentation data contents within the Media XML files.

9. Data queries and the query interface

The Media Query interface provides the means for querying the data contents of the Media CMS. All Media CMS data queries are performed exclusively through the query interface, which similarly to the update interface ensures that there is one well-defined route to data access with appropriate security checks. The Media Query interface is also the most significant component for the overall performance and scalability of the system.

Like the Media Update interface, the Media Query interface is technically a Java servlet which is accessed with HTTP POST requests. Both the input and output formats are XML-based with queries specified in the Media Query XML language and query results in the Media XML language, which is also the input format for the Media Update interface. This chapter describes the operating principles and rationale behind the Media Query implementation as well as the most significant challenges that were encountered, especially in regard to scalability.

9.1. Rationale for an XML query interface

There were a multitude of reasons for implementing a separate query interface instead of resorting to direct SQL queries to the database from the user interfaces and other components. There were a couple of alternatives for the actual implementation of the interface, of which a proprietary XML query language was chosen. The following subsections detail the rationale behind the choices that were made.

9.1.1. Advantages over plain SQL

A dedicated query interface has a number of advantages over plain SQL. These include the following:

- **Simplicity**. The database structures of the Media DB are quite complicated as the data contents of a single object are distributed among a number of tables, some of which have multiple rows for each object. Most queries also need to retrieve and combine a large number of objects which are connected by different kinds of relations. As a result, a single logical query from the user interfaces may actually result in a large number of complicated queries. A custom made query language, on the other, can be optimized to map closely to the structures of the data model instead of the physical storage structures, and can hence provide short and simple syntaxes for the common needs. The actual implementation can use highly complex SQL queries, but this complexity can be hidden from the user of the interface.
- **Security**. Direct SQL database access is subject to a number of security issues, such as SQL injection attacks [185]. A single poorly designed user interface implementation may therefore endanger the safety of the entire system, for example, by failing to perform the necessary input validation checks. A dedicated

query interface, however, can provide a well-defined database access point, which can centrally perform the necessary input validity and permission checks.

- **External integrations**. Direct database access is hardly an option for external integrations for a multitude of reasons, which include security and firewall restrictions. In practice these integrations require some sort of integration interfaces for data queries and updates.
- **Loose coupling from the exact database structures**. A dedicated query interface can not only hide the complexities of the actual database queries but also the internal structures of the database. This is important because the exact database structures are subject to change due to, for example, addition of new features or optimizations. With a query interface these changes affect only one component implementation, assuming the query format is kept the same.
- **Easier optimization**. Centralized database access provides also easier system-wide optimizations, as optimizations made to the query interface can boost the performance of all of the components that use it, with a single shot. The query interface can also implement more efficient caching mechanisms with caches that are shared between all of the components that are using it.

9.1.2. Query language alternatives

During the designing phase, there was one major alternative for direct SQL queries or to the implementation of a proprietary XML query interface. This alternative was XQuery [186], which, as a first though, seemed to be a good candidate for querying XML content. This initial intuition, however, was quite misleading, as the native data unit of the Media CMS data model is not actually an XML document, but an individual object, which can be connected to other objects in a number of ways. For this reason, XQuery had a number of drawbacks as, similarly to SQL, it doesn't map closely to the Media CMS data model, and the syntax for many common queries turned out to be quite tedious. There also weren't any readily available implementations that would be of use in a database based implementation, and hence it wouldn't have offered any time-saving shortcuts in the implementation.

9.2. The Media Query language

The Media Query language has some resemblance to SQL, but maps directly to the Media CMS data model. Because of this, most queries can be expressed with simple and short XML fragments. The following subsections contain an overview of the most significant features of the Media Query language. There are a number of other features, which have been omitted from this short overview.

9.2.1. Basic query structure

Figure 10 shows the basic query structure of the Media Query XML format. The structure is very similar to SQL containing the attributes that are selected, where

conditions, and the ordering specifications. The query is enclosed within the <mediaQuery> root element, which can contain any number of individual <select> query elements, which will be executed in order.

```
<mediaQuery>
      <select>
            <attributes>
                  …
            </attributes>
            <where>
                  …
            </where>
            <order>
                  …
            </order>
      </select>
</mediaQuery>
```

Figure 10: Basic Media Query XML structure

Possible contents for the <attributes>, <where>, and <order> elements include a number of elements which refer to individual properties of the objects. These include, for example, the <name>, <type> and <rdf> elements, which refer to the name, media type, and rdf properties, respectively. The <rdf> element can further specify the exact RDF properties that are targeted. These elements function a bit differently in different contexts. Within the <where> element they can specify the value to compare to, as well as the type of the comparison. Elsewhere they are simply empty elements which specify that the specified properties should be returned or used for ordering.

9.2.2. Selecting the returned data

The <attributes> element is used to specify the returned data contents. By default, an empty <attributes> element returns most of the basic media object properties but leaves out certain properties that require additional SQL queries. These properties, which include RDF-properties and source URLs, can be requested separately with <rdf> and <src> elements that are contained within the attributes element.

There are also some additional data returning elements that can be used in addition or instead of the <attributes> element. These include a number of aggregate query elements for returning the minimum, maximum, average, and count values for a specified data property. A single query can use any number of these elements and their results will be returned in order.

9.2.3. Where conditions

Where conditions can test a number of different properties, and conditions can be combined with <and>, <or>, and <not> elements, which signify logical operators. The query language also provides a convenient shortcut by automatically combining conditions that target the same property with the OR operator and others with the AND

operator, if the conditions are not inside explicit <and> or <or> elements. Experience has shown that most queries with several conditions can be written without ever explicitly specifying <and> or <or> elements.

9.2.4. Result ordering

The result order can be specified with <order> elements, which can specify a number of properties by which the result is ordered. The order can be either ascending or descending, which is selected by the asc attribute of the <order> element. Multiple <order> elements can be used for specifying both ascending and descending elements.

9.2.5. Queries through relations

The query language refers to the relations between media objects in two places. One of them is the <whereRelated> element, which can be used within the where conditions, and the other is the <selectRelated> element, which can be used inside attributes.

The <whereRelated> element can be used among other where conditions to test whether the object in question has related elements that fulfill the conditions that are defined inside the <whereRelated> element. This element has attributes which specify the axis and type of the required relations. The number of such relations can be restricted by the min and max attributes, so that only those objects will be matched for which the number of such relations is within these limits.

The <selectRelated> element can be used to select related objects, which will be returned as part of the result. These related objects are returned per each object on the outer select element (or outer nested <selectRelated> element). Similarly to the <whereRelated> element, this element can specify the axis and type of the relations. The allowed contents of the <selectRelated> element are the same as those of the <select> element.

The <selectRelated> element also supports a number of attributes for controlling how the related objects will be ordered, nested, and referenced within the result, as the Media XML format provides a number of possible arrangements as described in section 8.6. These options are beyond the scope of this overview.

9.2.6. Relation axes and virtual relations

Within the Media CMS data model, all physical media object relations are direct parent-child links between a source object and a target object. However, the Media CMS, and especially the Media Query language, also supports the notion of virtual relations, which are formed by combining or reversing physical relations. Virtual relations are based on the concept of relation axis, which is based on the XPath specification [187]. The supported relation axes are a subset of the XPath axes, and they include child, parent, ancestor, descendant, ancestor-or-self, and descendant-or-self axes.

These axes can be used, for example, within the `<whereRelated>` and `<selectRelated>` elements to specify, for instance, that the related objects must be descendants of the containing object. Optionally the `type` attribute can be used to specify that the entire chain of relations must be, for example, of type "component".

9.2.7. Media Query example

Figure 11 contains an example query in the Media Query XML language for which Figure 12 shows a possible response in the Media XML language.

```
<mediaQuery>
     <select>
          <attributes>
               <selectRelated axis="descendant">
                    <attributes>
                         <src />
                    </attributes>
                    <where>
                         <type>smilObject</type>
                    </where>
               </selectRelated>
          </attributes>
          <where>
               <type>article</type>
               <whereRelated axis="descendant">
                    <format>video/quicktime</format>
               </whereRelated>
          </where>
     </select>
</mediaQuery>
```

Figure 11: Media Query XML example

```
<mediaTransfer version="1.0">
     <group mediaType="article" id="1" name="Article1">
          <video             id="123"             name="Video1"
type="video/quicktime">
               <switch>
                    <video systemBitrate="500000">

     <src>http://10.10.10.10/v1_hi.qt</src>
                    </video>
                    <video>

     <src>http://10.10.10.10/v1_lo.qt</src>
                    </video>
               </switch>
          </video>
          <audio             id="125"             name="Audio1"
type="audio/quicktime">
               <src>http://10.10.10.10/a1.qt</src>
               <src>http://10.10.10.20/a1.qt</src>
          </audio>
     </group>
</mediaTransfer>
```

Figure 12: Example response for the query of Figure 11

This example selects objects of type “article” that have at least some video content in the QuickTime format and there happens to be only one such article in the response. The query also specifies that the results should include related descendant objects of type “smilObject” for each of the returned articles. The type condition utilizes the media type hierarchy and returns all objects that are of the specified type or its subtypes. These include, for example, the audio and video types, as well as switch objects, as is seen in the example result, but they do not include the tracks of the container formats, which are not of interested for this particular query. The example also demonstrates how the conditions and returned data contents for the related objects can be specified independently, as the `<whereRelated>` element tested for video content but the `<selectRelated>` element also allows audio content to the results.

The innermost `<attributes>` element also specifies that the source URLs should be returned in the results. Source URLs are one of those properties which are not returned by default unless explicitly asked. In the example response the audio object happens to contain a couple of alternative HTTP URLs for the actual audio file. These URLs are generated based on the information about the locations of the media files and configuration data about the delivery servers and their publishing points. These URLs can point to a number of different delivery servers, such as regular HTTP servers or streaming servers, and they can use different protocols, such as HTTP, MMS or RTSP.

9.3. XML to SQL mapping

The Media Query interface operates basically by converting the Media Query XML requests to SQL queries, whose results it converts back to Media XML responses. The XML requests can, however, contain a number of queries, which can also be arbitrarily complex, and hence a single XML request may be converted to a number of SQL queries. Typically each `<select>` and `<selectRelated>` element is translated to at least one SQL query. Oftentimes these queries are interdependent, as they use results of previous queries. Queries for `<selectRelated>` elements, for example, search for objects that are related to those objects that were selected by the containing `<select>` element. Basically this means that the object ids that were found by the outer query must be used as search conditions for performing the inner queries. This could be done by first retrieving the outer results and then including these results to the inner queries as IN conditions, for example. The query interface, however, tries to avoid unnecessary transfer of ids back and forth by creating temporary tables from the results of the queries, so that the intermediate results are stored locally inside the database, and these temporary tables can be directly joined to the queries that depend on their results. The temporary tables are also indexed which can improve the performance as compared to queries with IN conditions, for example.

The query interface works by executing first the queries that retrieve the ids of the objects that should be selected, and after these are found, it performs the queries that

select the data contents to be returned. The data contents are also retrieved for a block of objects (say, 1000 objects) at a time with a single query, which significantly reduces the number of SQL queries. The query interface also has caches that can be shared between the queries. This reduces the number of queries and data retrievals further.

9.4. Relational databases and hierarchical data

The DAG structure of the media objects and relations, and especially the hierarchical virtual axis reachability queries that need to be performed by the `<whereRelated>` and `<selectRelated>` elements, are quite challenging for the SQL query implementation, as recursive queries to hierarchical data structures are a well known weakness of relational databases and standard SQL, at least prior to the SQL99 standard [188]. The SQL99 standard contains the WITH clause which can be used for recursive queries but this clause is not yet widely supported, and MySQL, in particular, does not support it. Some databases have proprietary SQL extensions that can be used for recursive queries, such as Oracle's CONNECT BY clause [189]. MySQL, however, does not contain such features.

This was a significant problem for the Media Query implementation, as there were a lot of cases where recursive queries would be needed, but MySQL did not have any support for such queries. It was evident, that the only solution that could provide support for such queries, with reasonable performance, would be maintaining auxiliary tables that would provide the necessary structures for performing the hierarchic queries.

Four different query structures were considered in the design phase. These were transitive closures, nested sets, nested intervals, and materialized paths [190]. All of them can be used to represent tree structures and to perform reachability queries, and all of them have some advantages and weaknesses. The problem was, however, that the data model was a DAG and not a simple tree or a set of trees. Only transitive closures were able to directly represent such structures without major tweaking. Furthermore, analysis of the data contents of the first implemented services showed that the DAG was relatively shallow, with maximum path lengths less than 10 objects long, and the branching factors of the graphs were reasonably low. Hence it turned out that a transitive closure table, which would contain the entire transitive closure of the relation paths, would not grow too large. In fact, the row counts of such tables for the first services that were implemented turned out to be only about five times those of the direct relation tables. Hence transitive closures were chosen to be used for the implementation of the reachability queries.

Maintaining of transitive closures was, however, a significant problem, as the row counts were in the order of hundreds of thousands, and the updates needed to be relatively fast. Dong *et al.* [191] propose an algorithm for maintaining the transitive closure structure incrementally with SQL, but that algorithm turned out to be too slow

in practice. The problem was that this algorithm required queries which performed NOT EXISTS clauses and three-way self-joins on temporary tables that were very large for the data set in question, and the database just couldn't handle these with reasonable performance. To make things worse, self-joins on temporary tables is a well known [192] and long existed [193] limitation in MySQL.

The problem was ultimately solved with a seriously optimized database procedure, which recreated the entire transitive closure into temporary tables after each update batch (a set of inserts, updates, and deletes), and this temporary table was then compared to the permanent transitive closure table and only the differences were updated to the permanent table. The permanent table was not updated directly because the temporary table could use the much faster Memory storage engine with less indexing. This solution yielded good performance results, as the entire update sequence typically took only a few seconds or in some cases even a fraction of a second, with row counts in the order of hundreds of thousands.

9.5. Horizontal partitioning

Horizontal partitioning, also known as sharding, is the process of dividing different rows of a database table into multiple tables, which may be located in different databases. The rows can be partitioned, for example, by the value of one of the columns, so that the rows for which the column has values within some particular range are stored to a specific database instance. Partitioning can significantly improve performance and scalability, as the row counts of a single database can be kept within manageable limits. Partitioning is commonly utilized within the architectures of high volume online services. YouTube, for example, uses data partitioning based on the user ids, and these partitions are completely independent of each other [194]. [195]

Sharding could also be a highly useful feature for high-volume services that are implemented on top of the Media CMS platform. However, the highly interconnected DAG structure of the Media CMS data model does not lend itself easily to sharding. The problem is that, in principle, all of the objects in the database belong to a single DAG, and they can have arbitrary number of connections to other objects. Therefore the data contents are not easy to partition, in a generic way, into segments that would not be dependent on each other through their relations. Additionally, the data contents for a single object can be contained in several tables, all of which would need to be partitioned.

The data contents of the Media CMS could be divided into multiple databases on the Media Update interface implementation. This implementation could also duplicate some commonly used objects, such as categories, into more than one database, if required. Dividing the data contents into multiple databases is, however, only one side of the problem. The implementation of the Media Query implementation poses more

difficult problems. The problem is that the Media Query interface provides a wide range of queries that can retrieve interconnected objects, which could be partitioned into different databases. The Media Query implementation would need to execute individual XML queries into a number of different databases and then combine the results of those queries. This combination of results could be quite challenging though, as the results from the individual databases could be dependent on each other in complex ways.

The exact logic and partitioning rules are usually highly application specific, even if partitioning frameworks are used. Hibernate Shards [195], for example, provides some generalized sharding logic but requires application specific implementations for the details. This applies also in the case of the Media CMS, as while the underlying data model is generic and does not define any segmentations for the data contents, there may be natural service specific partitions in the service specific data models. Some services may, for example, have natural divisions based on the user id or they may have categorizations whose contents are completely independent on other categories. Hence Media CMS could implement the general portion of partitioning, but the detailed rules should be defined in a service specific way. Currently, however, sharding functionalities are not implemented.

9.6. Performance

Performance measurements of the Media Query interface have shown that the overhead induced by XML processing is quite negligible, and the vast majority of the execution times are spent on the actual SQL calls. These SQL calls, in turn, have been thoroughly analyzed in the context of the implemented services, and the generated queries have been found to be optimal in the sense that no further optimizations have been found, even if the queries were manually optimized. Overall, the performance that has been attained has been quite good, even though only a single database has been used. Therefore there hasn't been any pressure to utilize database clustering or to implement partitioning.

10. Summary

This thesis examined the current state of online video, various components and technologies that are related to online video services, and the implementation of such services. It suggested that while it is possible to implement online video services without understanding the low-level details of video formats, delivery mechanisms, and network protocols, basic knowledge of those issues can nevertheless be a significant benefit in making the right technological choices and in the fine-tuning of such services.

This thesis also introduced the Media CMS and described the design principles that were followed and the architectural decisions that were taken during the implementation of the system. Media CMS is intended as a generic platform for the implementation of different kinds of Internet audio and video services, although so far the main focus on the development has been on B2C services that feature exclusively professionally created content. This thesis described how the current system relates to the online video service types which were described in chapter 3 and how it can utilize off-the-shelf components, such as those that were described in chapter 6, for the delivery and playback of media files, and also how it takes into account different video formats and delivery methods, such as those that were described in chapters 4 and 5, respectively.

This thesis also described some particular challenges that were encountered during the implementation of the system. Perhaps the most significant of these was a non-technical one, the tight schedule of the project, which was being tackled with design choices that tried to conserve some time with the maximization of code reuse and with the use of of-the-shelf components and standard formats. The most significant technical problems were closely related to the Media Query interface, which is probably the most important component of the system. These included reachability queries to hierarchical data in a relational database and horizontal partitioning of highly interconnected data. The problem of hierarchical queries was solved with the use of transitive closure tables, as it was the only solution which was directly suitable for the representation of DAG structures, and it also offered good performance. The horizontal partitioning problem is still an open issue, although not an urgent one, as so far the performance of the system has been sufficient without partitioning.

References

(All online links retrieved on May 15, 2008)

[1] Brooks Barnes, Big TV's Broadband Blitz, The Wall Street Journal Online, August 1, 2006. Available as http://wsjclassroomedition.com/archive/06sep/06sep_bigtvsbroadbandblitz.pdf

[2] Silicon Valley / San Jose Business Journal, Study: Broadband 'killer app' found, February 3, 2004. Available as http://sanjose.bizjournals.com/sanjose/stories/2004/02/02/daily22.html

[3] John Earnhardt, Video on Demand: The Next Killer App?, Cisco High Tech Policy Blog, April 9, 2005. Available as http://blogs.cisco.com/gov/2005/04/video_on_demand_the_next_kille.html

[4] Ted Hanss, Digital Video: Internet2 Killer App or Dilbert's Nightmare?, *EDUCAUSE Review*, May/June 2001. Available as http://www.educause.edu/ir/library/pdf/erm0130.pdf

[5] YouTube web site, Available as http://youtube.com/

[6] Alexa web site. Available as http://www.alexa.com/

[7] Lev Grossman, Best Invention: YouTube, Time.com. Available as http://www.time.com/time/2006/techguide/bestinventions/inventions/youtube.html

[8] Google Press Center, Google To Acquire YouTube for $1.65 Billion in Stock, October 9, 2006. Available as http://www.google.com/press/pressrel/google_youtube.html

[9] Cisco Systems, Inc., The Exabyte Era, White Paper, January 14, 2008. Available as http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/net_implementation_white_paper0900aecd806a81a7.pdf

[10] The Economist, Overdoing it?, June 7, 2007. Available as http://www.economist.com/science/tq/displaystory.cfm?story_id=9249193

[11] Grant Gross, Internet Could Max Out in 2 Years, Study Says, PC World, November 24, 2007. Available as http://www.pcworld.com/article/id,139885-c,researchreports/article.html

[12] Martin LaMonica, Experts: No stopping flood of Web video, CNET News.com, February 7, 2007. Available as http://www.news.com/2100-1025_3-6157283.html

[13] BBC News, Online video 'eroding TV viewing', November 27, 2006. Available as http://news.bbc.co.uk/1/hi/entertainment/6168950.stm

[14] Eryn Brown, Video Unlimited, *Wired* Issue 14.07, July 2006. Available as http://www.wired.com/wired/archive/14.07/video.html

[15] The Nielsen Company, 81 Million People in U.S. Watch Broadband Video at Home or Work, According to Nielsen and CTAM, Press Release, July 17, 2007. Available as http://www.nielsen.com/media/pr_070717.html

[16] Mary Madden, Online Videos Go Mainstream, Pew Research Center, Pew Internet & American Life Project, July 25, 2007. Available as http://pewresearch.org/pubs/552/online-videos-go-mainstream

[17] comScore, Inc., YouTube Continues to Lead U.S. Online Video Market with 28 Percent Market Share, According to comScore Video Metrix, Press Release, November 30, 2007. Available as http://www.comscore.com/press/release.asp?press=1929

[18] Tilastokeskus, Internetin käyttötarkoitukset keväällä 2007, prosenttia internetin käyttäjistä, September 28, 2007. Available as http://www.stat.fi/til/sutivi/2007/sutivi_2007_2007-09-28_tau_001.html

[19] Tietokone.fi, Suomen nettiradiot avautuvat taas, June 27, 2007. Available as http://www.tietokone.fi/uutta/uutinen.asp?news_id=30845

[20] Ellacoya Networks Inc., Ellacoya Data Shows Web Traffic Overtakes Peer-to-Peer (P2P) as Largest Percentage of Bandwidth on the Network, NXTcomm Show, Chicago, June 18, 2007. Available as http://www.ellacoya.com/news/pdf/2007/NXTcommEllacoyaMediaAlert.pdf

[21] Andrew Parker, Addressing the cost and performance challenges of digital media content delivery, P2P Media Summit Presentation, October 23, 2006. Available as http://www.dcia.info/activities/p2pmsla2006/CacheLogic.ppt

[22] Andrew Donoghue, AT&T: Internet to hit full capacity by 2010, CNET News.com, April 18, 2008. Available as http://www.news.com/ATT-Internet-to-hit-full-capacity-by-2010/2100-1034_3-6237715.html

[23] BBC News, Net firm warns on web video costs, August 13, 2007. Available as http://news.bbc.co.uk/2/hi/technology/6944176.stm

[24] Bobby White, Its Creators Call Internet Outdated, Offer Remedies, *The Wall Street Journal*, October 2, 2007. Available as http://online.wsj.com/article/SB119128309597345795.html

[25] Marguerite Reardon, Industry watch Toolkit - Video demand helps Cisco profits soar, CNET News.com, February 7, 2007. Available as http://news.zdnet.co.uk/itmanagement/0,1000000308,39285822,00.htm

[26] Joel Waldfogel, Lost on the Web: Does Web Distribution Stimulate or Depress Television Viewing?, Preliminary paper, The Wharton School, University of Pennsylvania, August 10, 2007. Available as http://bpp.wharton.upenn.edu/waldfogj/pdfs/Lost_on_the_web.pdf

[27] Accenture, User-Generated Content Is Top Threat to Media and Entertainment Industry, Accenture Survey Finds, April 16, 2007. Available as http://accenture.tekgroup.com/article_display.cfm?article_id=4534

[28] Greg Sandoval, Local TV stations face Net threat, CNET News.com, April 17, 2007. Available as http://www.news.com/Local-TV-stations-face-Net-threat/2100-1025_3-6176561.html

[29] Interactive Advertising Bureau, IAB Internet Advertising Revenue Report conducted by PricewaterhouseCoopers (PWC), Reports for years 2002-2007. Available as http://www.iab.net/insights_research/1357

[30] eMarketer Inc., Online Ad Spending to Total $19.5 Billion in 2007, Press Release, February 28, 2007. Available as http://www.emarketer.com/Article.aspx?id=1004635

[31] David Hallerman, Internet Video: Advertising Experiments and Exploding Content, eMarketer Report summary, November 2006. Available as http://www.emarketer.com/Reports/All/Em_video_internet_nov06.aspx

[32] Jakob Nielsen, Jakob Nielsen's Alertbox for April 5, 1998: Nielsen's Law of Internet Bandwidth, April 5, 1998. Available as http://www.useit.com/alertbox/980405.html

[33] CU-SeeMe Development Team, The CU-SeeMe Project. Available as http://myhome.hanafos.com/~soonjp/project.html

[34] WXYC homepage, WXYC's Simulcast – WXYC's groundbreaking internet simulcast is now 10 years old!. Available as http://www.wxyc.org/about/first/

[35] ABC's "World News Now" Makes Broadcasting History, November 25, 1995. Available as http://scout.wisc.edu/Projects/PastProjects/NH/95-11/95-11-27/0015.html

[36] RealNetworks, Inc., RealNetworks wins Emmy Award for pioneering Internet streaming technology, Press Release, January 8, 2007. Available as http://www.realnetworks.com/company/press/releases/2007/emmy.html

[37] Timothy Hart, KCTU-TV earns a place in television, Internet history, *Wichita Business Journal*, February 13, 1998. Available as http://wichita.bizjournals.com/wichita/stories/1998/02/16/focus1.html

[38] CinemaNow.com, Company background, 2008. Available as http://www.cinemanow.com/Aboutus-Background.aspx

[39] BBC News, Web's first 'e-première', August 12, 2003. Available as http://news.bbc.co.uk/2/hi/entertainment/3144775.stm

[40] MTV New Media, Paramount Pictures Digital Entertainment, BLOCKBUSTER, Hollywood's First Studio-Backed Broadband Movie Streamed In Its Entirety For Free, Press release, December 13, 2007. Available as http://www.jackassworld.com/blog/2007/12/12/mtv-new-media-paramount-pictures-digital-entertainment-and-blockbuster-unleash-jackass-25/

[41] Tim O'Reilly, What Is Web 2.0 - Design Patterns and Business Models for the Next Generation of Software, O'Reilly Media, Inc, September 30, 2005. Available as http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html

[42] Tim O'Reilly, Web 2.0 Compact Definition: Trying Again, O'Reilly radar, O'Reilly Media Inc., December 10, 2006. Available as http://radar.oreilly.com/archives/2006/12/web_20_compact.html

[43] Scott Laningham (ed.), developerWorks Interviews: Tim Berners-Lee, IBM developerWorks, August 22, 2006. Available as http://www.ibm.com/developerworks/podcast/dwi/cm-int082206txt.html

[44] Google Inc., YouTube APIs and Tools, Google Code, 2008. Available as http://code.google.com/apis/youtube/overview.html

[45] Danah M. Boyd and Nicole B. Ellison, Social Network Sites: Definition, History, and Scholarship, *Journal of Computer-Mediated Communication*, 2008. Available as http://www.blackwell-synergy.com/doi/pdf/10.1111/j.1083-6101.2007.00393.x

[46] Martin Halvey and Mark T. Keane, Exploring Social Dynamics in Online Media Sharing, *16th International World Wide Web Conference (WWW2007)*, May 8-12, 2007. Available as http://www2007.org/posters/poster976.pdf

[47] Kristina Lerman, Social Browsing & Information Filtering in Social Media, University of Southern California, Information Sciences Institute, February 2, 2008. Available as http://arxiv.org/PS_cache/arxiv/pdf/0710/0710.5697v1.pdf

[48] Dailymotion web site. Available as http://www.dailymotion.com/

[49] Megavideo web site. Available as http://www.megavideo.com/

[50] Veoh web site. Available as http://www.veoh.com/

[51] YouTube, LLC, YouTube Company History, Available as http://youtube.com/t/about

[52] YouTube.com, You Choose '08. Available as http://www.youtube.com/youchoose

[53] BBC News, Pakistan blocks YouTube website, February 24, 2008. Available as http://news.bbc.co.uk/2/hi/south_asia/7261727.stm

[54] Disney-ABC Television Group, Disney-ABC Television Group takes ABC primetime online, offering hit shows on ABC.com during May and June, Press Release, April 10, 2006. Available as http://corporate.disney.go.com/news/corporate/2006/2006_0410_abchitshows.html

[55] Disney-ABC Television Group, Disney-ABC Television Group's Emmy-winning ABC.com brings back enhanced, ad-supported broadband player this month, Press Release, September 13, 2006. Available as http://corporate.disney.go.com/news/corporate/2006/2006_0913_abcbroadband.html

[56] Disney-ABC Television Group, Disney-ABC and Warner Bros. announce groundbreaking digital distribution agreement, Press Release, September 6, 2007. Available as http://www.disneyabctv.com/web/NewsRelease/DispDNR.aspx?id=090607_07

[57] National Association of Broadcasters (NAB), High definition Internet streaming video, NAB TV TechCheck, July 30, 2007. Available as http://www.nab.org/xert/scitech/pdfs/tv073007.pdf

[58] Hulu web site, Available as http://www.hulu.com/

[59] BBC iPlayer web site, Available as http://www.bbc.co.uk/iplayer/

[60] BBC Press Office, BBC enters strategic relationship with Adobe to enhance BBC iPlayer and bbc.co.uk, Press Release, October 16, 2007. Available as http://www.bbc.co.uk/pressoffice/pressreleases/stories/2007/10_october/16/adobe.shtml

[61] BBC Press Office, Ashley Highfield appointed as CEO of Kangaroo, Press Release, April 14, 2008. http://www.bbc.co.uk/pressoffice/bbcworldwide/worldwidestories/pressreleases/2008/04_april/ashley_highfield_kangaroo.shtml

[62] YLE Areena web site. Available as http://areena.yle.fi/

[63] Petro Poutanen, Yle uittaa verkkoon kaikki ohjelmat, Digitoday, July 4, 2007, Available as http://www.digitoday.fi/viihde/2007/07/04/Yle+uittaa+verkkoon+kaikki+ohjelmat/200716378/66

[64] Katri Kallionpää, Yleisradio rajaa verkkopalveluaan tv-maksun maksaneille, Helsingin Sanomat, October 10, 2007. Available as http://www.hs.fi/talous/artikkeli/Yleisradio+rajaa+verkkopalveluaan+tv-maksun+maksaneille/1135230948793

[65] MTV3 Netti-tv web site. Available as http://www.mtv3.fi/nettitv/

[66] Nelonen NettiTV web site. Available as http://www.nelonen.fi/nettitv/

[67] Miro web site web site. Available as http://www.getmiro.com/

[68] Vuze web site web site. Available as http://www.vuze.com/

[69] Bram Cohen, The BitTorrent Protocol Specification, BitTorrent.org, January 10, 2008. Available as http://www.bittorrent.org/beps/bep_0003.html

[70] Joost web site. Available as http://www.joost.com/

[71] Babelgum web site. Available as http://www.babelgum.com/

[72] TVU networks web site. Available as http://www.tvunetworks.com/

[73] SopCast web site. Available as http://www.sopcast.org/

[74] mariposaHD.tv web site. Available as http://www.mariposahd.tv/

[75] Koeajo.tv web site. Available as http://www.koeajo.tv/

[76] Amy Harris and Greg Ireland, Enabling IPTV : What Carriers Need to Know to Succeed, IDC White Paper, May 2005. Available as http://www.emc.com/collateral/analyst-reports/idc-iptv-whitepaper-jun-9-05.pdf

[77] Jeremy Allaire, IPTV vs. "Internet of Video", Brightcove presentation, March 10, 2005. Available as http://breeze.brightcove.com/p47258018/

[78] ITU Telecommunication Standardization Sector (ITU-T) official site. Available as http://www.itu.int/ITU-T/

[79] Nejat Kamaci and Yucel Altunbasak, Performance comparison of the emerging H.264 video coding standard with the existing standards, *IEEE International Conference on Multimedia and Expo*, July 2003. Available as http://www.ece.gatech.edu/research/labs/MCCL/pubs/dwnlds/h26l_analysis.pdf

[80] Michael Horowitz, Anthony Joch, Faouzi Kossentini, and Antti Hallapuro, H.264/AVC Baseline Profile Decoder Complexity Analysis, *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 13, No. 7, July 2003. Available as http://lts4www.epfl.ch/teaching/ic/repository/01218201.pdf

[81] Apple Inc., QuickTime File Format Specification, September 4, 2007. Available as http://developer.apple.com/documentation/QuickTime/QTFF/qtff.pdf

[82] Thomas Wiegand, Gary J. Sullivan, Gisle Bjøntegaard, and Ajay Luthra, Overview of the H.264/AVC Video Coding Standard, *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 13, No. 7, July 2003. Available as http://www.cs.ubc.ca/~krasic/cpsc538a-2005/papers/h264avc-overview.pdf

[83] International Telecommunication Union (ITU), CCITT, Recommendation T.81, September 18, 1992. Available as http://www.w3.org/Graphics/JPEG/itu-t81.pdf

[84] Jens-Rainer Ohm and Gary Sullivan (eds.), Introduction to MPEG-4 Advanced Video Coding, ISO/IEC JTC1/SC29/WG11, July 2005. Available as http://www.chiariglione.org/mpeg/technologies/mp04-avc/index.htm

[85] Huahui Wu, Mark Claypool, and Robert Kinicki, Guidelines for Selecting Practical MPEG Group of Pictures, *Proceedings of IASTED International Conference on Internet and Multimedia Systems and Applications (EuroIMSA)*, February 2006. Available as http://web.cs.wpi.edu/~claypool/papers/practical-gop/vGOP.pdf

[86] Microsoft Corporation, Encoding Audio and Video with Windows Media Codecs, October 2007. Available as http://www.microsoft.com/windows/windowsmedia/howto/articles/codecs.aspx

[87] H. Koumaras, G. Gardikis, A. Kourtis, and D. Martakos, Quantitative Perceptual Comparison of Variable Bit Rate over Constant Bit Rate Encoding Scheme for MPEG-4 Video, *Journal of Electronic Imaging (JEI), SPIE, IS&T*, July 2007. Available as http://aias.iit.demokritos.gr/~koumaras/Quantitative%20Perceptual%20Comparison%20of%20VBR%20over%20CBR%20Encoding%20Scheme%20for%20MPEG-4%20Video.pdf

[88] Subhabrata Sen, Jennifer L. Rexford, Jayanta K. Dey, James F. Kurose, and Donald F. Towsley, Online Smoothing of Variable-Bit-Rate Streaming Video, *IEEE Transactions on Multimedia*, Vol. 2, No. 1, March 2000. Available as http://www.cmlab.csie.ntu.edu.tw/~pkhsiao/PDF/Online%20smoothing%20of%20variable-bit-rate%20streaming%20video.pdf

[89] Bill Birney, Intelligent Streaming, Microsoft Corporation article, May 2003. Available as http://www.microsoft.com/windows/windowsmedia/howto/articles/intstreaming.aspx

[90] ISO/IEC Moving Picture Experts Group (MPEG) working group official site. Available as http://www.chiariglione.org/mpeg/

[91] Gary J. Sullivan, Overview of International Video Coding Standards (preceding H.264/AVC), ITU-T VICA Workshop, July 22-23, 2005. Available as http://www.itu.int/ITU-T/worksem/vica/docs/presentations/S0_P2_Sullivan.pdf

[92] Mohammed Ghanbari, Video coding: past, present and future, Ingenia Magazine, Article - Issue 7, February 2001. Available as http://www.ingenia.org.uk/ingenia/articles.aspx?Index=96

[93] J. Ott, C. Bormann, G. Sullivan, S. Wenger, and R. Even (ed.), RTP Payload Format for ITU-T Rec. H.263 Video, RFC 4629, January 2007. Available as http://tools.ietf.org/html/rfc4629

[94] Stefano Polidori, Multimedia Services from Narrowband to Broadband, ITU Telecommunication Standardization Workshop, October 25-27, 2006. Available as http://www.itu.int/dms_pub/itu-t/oth/06/06/T060600000040001PDFE.pdf

[95] ISO/IEC JTC1/SC29/WG11, Report of The Formal Verification Tests on AVC (ISO/IEC 14496-10 | ITU-T Rec. H.264), December 2003. Available as http://www.chiariglione.org/mpeg/working_documents/mpeg-04/avc/avc_vt.zip

[96] Ajay Luthra, MPEG-4 AVC/H.264 Digital Video Compression Standard, Joint ITU-T Workshop and IMTC Forum 2006, May 9-11, 2006. Available as http://www.itu.int/ITU-T/worksem/h325/200605/presentations/s3p1-luthra.pdf

[97] Detlev Marpe, Thomas Wiegand, and Gary J. Sullivan, The H.264/MPEG4 Advanced Video Coding Standard and its Applications, IEEE Communications Magazine, August 2006. Available as http://iphome.hhi.de/wiegand/assets/pdfs/h264-AVC-Standard.pdf

[98] Jay Loomis and Mike Wasson, VC-1 Technical Overview, Microsoft Corporation articles, October 2007. Available as http://www.microsoft.com/windows/windowsmedia/howto/articles/vc1techoverview.aspx

[99] RealNetworks, Inc., RealVideo 10 Technical Overview, Version 1.0, 2003. Available as http://docs.real.com/docs/rn/rv10/RV10_Tech_Overview.pdf

[100] On2 Technologies web site. Available as http://www.on2.com/

[101] On2 Technologies, Inc., Advantages of TrueMotion VP6 Technology, White Paper, February 17, 2004. Available as http://multimedia.cx/mirror/vp6-white-paper.pdf

[102] On2 Technologies, Inc., TrueMotion VP7 Video Codec, White Paper, January 10, 2005. Available as http://multimedia.cx/mirror/vp7-white-paper.pdf

[103] Adobe Systems Incorporated, Adobe Flash Media Server 3, White Paper, 2008. Available as http://www.adobe.com/products/flashmediaserver/pdfs/FlashMediaServer3_WhitePaper_ue.pdf

[104] On2 Technologies, Inc., On2 Technologies Helps Provide High Quality Video for Skype, Press Release, November 12, 2007. Available as http://www.on2.com/index.php?id=439&news_id=581

[105] Xiph.org Foundation, Theora video compression web site. Available as http://theora.org/

[106] BBC Research, Dirac web site. Available as http://www.bbc.co.uk/rd/projects/dirac/index.shtml

[107] Peter Meerwald, The Dirac Video Codec, Department of Computer Science, University of Salzburg, April 18, 2007. Available as http://www.cosy.sbg.ac.at/~pmeerw/Compression/pv0704/dirac.pdf

[108] Microsoft Corporation, Advanced Systems Format (ASF) Specification, December 2004. Available as http://www.microsoft.com/windows/windowsmedia/forpros/format/asfspec.aspx

[109] ITU-T, H.222.0: Information technology - Generic coding of moving pictures and associated audio information: Systems, Recommendation H.222.0, May 2006. Available as http://www.itu.int/rec/T-REC-H.222.0-200605-I/en

[110] MPEG-4 Industry Forum, MPEG-4 – The Media Standard, November 19, 2002. Available as http://www.m4if.org/public/documents/vault/m4-out-20027.pdf

[111] Apple Computer Inc., MPEG-4 - The new standard for multimedia on the Internet, powered by QuickTime, Fact Sheet, 2003. Available as http://images.apple.com/quicktime/pdf/MPEG4_v3.pdf

[112] 3GPP Organizational Partners, 3GPP TS 26.244 V7.3.0 (2007-12), Technical Specification, 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects Transparent end-to-end packet switched streaming service (PSS); 3GPP file format (3GP) (Release 7), December 2007. Available as http://www.3gpp.org/ftp/Specs/html-info/26244.htm

[113] Microsoft Corporation, AVI RIFF File Reference, Windows Media Developer Center, 2008. Available as http://msdn2.microsoft.com/en-us/library/ms779636(VS.85).aspx

[114] RealNetworks, Inc., Helix Client and Server SDK (r5 draft), Appendix E: RealMedia File Format (RMFF) Reference, 2005. Available as https://common.helixcommunity.org/2003/HCS_SDK_r5/htmfiles/rmff.htm

[115] Adobe Systems Incorporated, ActionScript 2.0 Language Reference – Video, 2008. Available as http://livedocs.adobe.com/flash/9.0/main/wwhelp/wwhimpl/common/html/wwhelp.htm?context=LiveDocs_Parts&file=00002303.html

[116] Matroska.org, Matroska web site. Available as http://www.matroska.org/

[117] Xiph.Org, The Ogg container format, 2007. Available as http://www.xiph.org/ogg/

[118] Martin Nilsson, Extensible Binary Markup Language, Draft, March 15, 2004. Available as http://www.matroska.org/technical/specs/rfc/ebml-1.0.txt

[119] Apple Computer Inc., QuickTime Overview, August 11, 2005. Available as http://developer.apple.com/documentation/QuickTime/RM/Fundamentals/QTOverview/QTOverview.pdf

[120] Manitu Group, FLV MetaData Injector web site. Available as http://www.buraks.com/flvmdi/

[121] xmoov.com, xmoov-php - HTTP pseudo-streaming script, 2007. Available as http://xmoov.com/xmoov-php/

[122] Jon Postel (ed.), Internet Protocol - DARPA Internet program - Protocol specification, RFC 791, September 1981. Available as http://tools.ietf.org/html/rfc791

[123] S. Deering and R. Hinden, Internet Protocol, Version 6 (IPv6) Specification, RFC 2460, December 1998. Available as http://tools.ietf.org/html/rfc2460

[124] R. Hinden and S. Deering, IP Version 6 Addressing Architecture, RFC 4291, February 2006. Available as http://tools.ietf.org/html/rfc4291

[125] Jeffrey Mogul, Broadcasting Internet Datagrams in the Presence of Subnets, RFC 922, October 1984. Available as http://tools.ietf.org/html/rfc922

[126] S. Deering, Host Extensions for IP Multicasting, RFC 1112, August 1989. Available as http://tools.ietf.org/html/rfc1112

[127] H. Holbrook and B. Cain, Source-Specific Multicast for IP, RFC 4607, August 2006. Available as http://tools.ietf.org/html/rfc4607

[128] Sylvia Ratnasamy, Andrey Ermolinskiy, and Scott Shenker, Revisiting IP Multicast, SIGCOMM'06, September 11–15, 2006. Available as http://berkeley.intel-research.net/sylvia/frm-sigcomm06.pdf

[129] B. Quinn and K. Almeroth, IP Multicast Applications: Challenges and Solutions, RFC 3170, September 2001. Available as http://tools.ietf.org/html/rfc3170

[130] Jon Postel (ed.), Transmission Control Protocol, DARPA Internet Program Protocol Specification, RFC 793, September 1981. Available as http://tools.ietf.org/html/rfc793

[131] J. Postel, User Datagram Protocol, RFC 768, August 28, 1980. Available as http://tools.ietf.org/html/rfc768

[132] E. Kohler, M. Handley, and S. Floyd. Datagram Congestion Control Protocol (DCCP), RFC 4340, March 2006. Available as http://tools.ietf.org/html/rfc4340

[133] R. Stewart (ed.), Stream Control Transmission Protocol, RFC 4960, September 2007. Available as http://tools.ietf.org/html/rfc4960

[134] RealNetworks, Inc., Helix Server Administration Guide - Helix Server Version 11.1, September 20, 2007. Available as http://docs.real.com/docs/server11/wireline/HelixServerAdmin.pdf

[135] M. Allman, V. Paxson, and W. Stevens, TCP Congestion Control, RFC 2581, April 1999. Available as http://tools.ietf.org/html/rfc2581

[136] Sunand Tullimalli, Multimedia Streaming Using Multiple TCP Connections, Master of Science thesis, Oregon State University, September 6, 2006. Available as http://ir.library.oregonstate.edu/dspace/bitstream/1957/3148/1/thesis.pdf

[137] Jae Chung and Mark Claypool, Empirical Evaluation of the Congestion Responsiveness of RealPlayer Video Streams, *Kluwer Multimedia Tools and Applications*, Volume 31, Number 2, November 2006. Available as http://web.cs.wpi.edu/~claypool/papers/h2h-journal/h2h.pdf

[138] H. Schulzrinne, A. Rao, and R. Lanphier, Real Time Streaming Protocol (RTSP), RFC 2326, April 1998. Available as http://tools.ietf.org/html/rfc2326

[139] M. Handley, V. Jacobson, and C. Perkins, SDP: Session Description Protocol, RFC 4566, July 2006. Available as http://tools.ietf.org/html/rfc4566

[140] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, RTP: A Transport Protocol for Real-Time Applications, RFC 3550, July 2003. Available as http://tools.ietf.org/html/rfc3550

[141] H. Schulzrinne and S. Casner, RTP Profile for Audio and Video Conferences with Minimal Control, RFC 3551, July 2003. Available as http://tools.ietf.org/html/rfc3551

[142] S. Wenger, M.M. Hannuksela, T. Stockhammer, M. Westerlund, and D. Singer, RTP Payload Format for H.264 Video, RFC 3984, February 2005. Available as http://tools.ietf.org/html/rfc3984

[143] Microsoft Corporation, Windows Media Services web site. http://www.microsoft.com/windows/windowsmedia/forpros/server/server.aspx

[144] RealNetworks Inc., RealNetworks Media Servers web site. Available as http://www.realnetworks.com/products/media_delivery.html

[145] Apple Inc., QuickTime Streaming Server web site. Available as http://www.apple.com/quicktime/streamingserver/§

[146] Apple Inc., Open Source Streaming Server web site. Available as http://developer.apple.com/opensource/server/streaming/index.html

[147] Adobe Systems Inc., Adobe Flash Media Server products web site. Available as http://www.adobe.com/products/flashmediaserver/

[148] Apple Computer Inc., QuickTime Streaming Server/Darwin Streaming Server Administrator's Guide, 2002. Available as http://developer.apple.com/opensource/server/streaming/qtss_admin_guide.pdf

[149] Lei Guo, Enhua Tan, Songqing Chen, Zhen Xiao, Oliver Spatscheck, and Xiaodong Zhang, Delving into Internet Streaming Media Delivery: A Quality and Resource Utilization Perspective, *Proceedings of Internet Measurement Conference (ACM SIGCOMM/USENIX IMC 2006)*, October 25-27, 2006. Available as http://www.cs.gmu.edu/~sqchen/publications/streamQ-imc2006.pdf

[150] Microsoft Corporation, Streaming Media Services, Windows Server 2008 Technical Library, Microsoft TechNet. Available as http://technet2.microsoft.com/windowsserver2008/en/library/40f7b266-fc87-4742-bbd1-496f104761011033.mspx

[151] Alexandre Ferreira, Optimizing Microsoft Windows Media Services 9 Series, Microsoft Windows Digital Media Division, March 2005. Available as http://download.microsoft.com/download/8/f/4/8f47ea6a-254a-421b-9542-c3e5965c8be7/optimize_web.doc

[152] Apple Inc., QuickTime Streaming, 2008. Available as http://www.apple.com/quicktime/technologies/streaming/

[153] Wowza Media Systems, Wowza Media Server Pro Unlimited product pages. Available as http://www.wowzamedia.com/products.html

[154] Red5 : Open Source Flash Server web site. Available as http://www.osflash.org/red5

[155] VideoLAN web site. Available as http://www.videolan.org/

[156] MPlayer web site. Available as http://www.mplayerhq.hu/design7/news.html

[157] Microsoft Corporation, Windows Media Player web site. http://www.microsoft.com/windows/windowsmedia/player/default.aspx

[158] RealNetworks, RealPlayer web site. Available as http://europe.real.com/player/

[159] Opera Software, Installation of Plug-ins for Opera on Linux, 2008. Available as http://www.opera.com/linux/docs/plugins/install

[160] Apple Inc., QuickTime Player web site. Available as http://www.apple.com/quicktime/player/

[161] Adobe Systems Incorporated, Adobe Flash Player web site. Available as http://www.adobe.com/products/flashplayer/

[162] Microsoft Corporation, Microsoft Silverlight web site. Available as http://www.microsoft.com/silverlight/

[163] Adobe Systems Incorporated, Flash Player Penetration. Available as http://www.adobe.com/products/player_census/flashplayer/

[164] Microsoft Corporation, Windows Media Player multimedia file formats, Microsoft Help and Support, May 12, 2008 Available as http://support.microsoft.com/kb/316992

[165] Apple Inc., QuickTime Streaming Server Modules Programming Guide, April 29, 2005. Available as http://developer.apple.com/documentation/QuickTime/QTSS/QTSS.pdf

[166] MozillaZine web site, Full Step-By-Step Guide: Embedded Windows Media in Firefox, January 23, 2005. Available as http://forums.mozillazine.org/viewtopic.php?t=206213

[167] Microsoft Corporation Port 25 web site, Windows Media Player Plug-in for Firefox, April 16, 2007. Available as http://port25.technet.com/archive/2007/04/16/windows-media-player-plug-in-for-firefox.aspx

[168] Tim Statler, Using video in Macromedia Flash MX, Macromedia Flash Support Center, May 21, 2002. Available as http://www.adobe.com/support/flash/images_video/flash_video/

[169] RealNetworks Inc., RealPlayer Help, 2007. Available as http://i.realone.com/product/help/rp11/en/R1P.htm

[170] The World Wide Web Consortium (W3C), Synchronized Multimedia Integration Language (SMIL 2.1), W3C Recommendation, December 13, 2005. Available as http://www.w3.org/TR/2005/REC-SMIL2-20051213/

[171] RealNetworks Inc., RealNetworks Production Guide With RealPlayer 10, July 20, 2004. Available as http://service.real.com/help/library/guides/ProductionGuide/prodguide/realpgd.htm

[172] Apple Computer, Inc., Interactive Movies, October 1, 2002. Available as http://developer.apple.com/documentation/QuickTime/IQ_InteractiveMovies/insideqt_intmov.pdf

[173] Apple Inc., QuickTime 7.1.3 and Adobe Flash, December 11, 2007. Available as http://docs.info.apple.com/article.html?artnum=304341

[174] Apple Computer, Inc., SMIL Scripting Guide for QuickTime, June 4, 2005. Available as http://developer.apple.com/documentation/QuickTime/Conceptual/QTScripting_SMIL/QTScripting_SMIL.pdf

[175] Apple Inc., QuickTime Media Skins, 2008. Available as http://www.apple.com/quicktime/technologies/mediaskins/

[176] MySQL web site. Available as http://www.mysql.com/

[177] Unicode, Inc., Unicode Home Page. Available as http://www.unicode.org/

[178] Erich Gamma, Richard Helm, Ralph Johnson, and John M. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley Professional, 1994.

[179] RSS Advisory Board, RSS 2.0 Specification, version 2.0.10, October 15, 2007. Available as http://www.rssboard.org/rss-specification

[180] Colm MacCarthaigh, Joost Network Architecture, April 4, 2007. Available as http://www.layer3media.com/joost/joost-network.pdf

[181] Microsoft Corporation, Microsoft Windows CE .NET 4.2 - ASX Elements Reference, Windows Embedded Developer Center, April 13, 2005. Available as http://msdn.microsoft.com/en-us/library/ms910265.aspx

[182] The World Wide Web Consortium (W3C), RDF/XML Syntax Specification (Revised), W3C Recommendation, February 10, 2004. Available as http://www.w3.org/TR/rdf-syntax-grammar/

[183] The World Wide Web Consortium (W3C), Resource Description Framework (RDF), 2008. Available as http://www.w3.org/RDF/

[184] The Dublin Core Metadata Initiative web site. Available as http://dublincore.org/

[185] The Open Web Application Security Project (OWASP), SQL Injection, December 14, 2007. Available as http://www.owasp.org/index.php/SQL_injection

[186] The World Wide Web Consortium (W3C), XQuery 1.0: An XML Query Language, W3C Recommendation, January 23, 2007. Available as http://www.w3.org/TR/xquery

[187] The World Wide Web Consortium (W3C), XML Path Language (XPath) 2.0, W3C Recommendation, January 23, 2007. Available as http://www.w3.org/TR/xpath20/

[188] Rajasekar Krishnamurthy, Venkatesan T. Chakaravarthy, Raghav Kaushik, and Jeffrey F. Naughton, Recursive XML Schemas, Recursive XML Queries, and Relational Storage: XML-to-SQL Query Translation, *Proceedings of ICDE 2004*. Available as http://pages.cs.wisc.edu/~naughton/includes/papers/recursiveQueryTranslation.pdf

[189] Oracle Corporation, Oracle Database, SQL Language Reference, 11g Release 1 (11.1), B28286-03, May 2008. Available as http://download.oracle.com/docs/cd/B28359_01/server.111/b28286.pdf

[190] Vadim Tropashko, Trees in SQL: Nested Sets and Materialized Path, DBAzine.com, April 13, 2005. Available as http://www.dbazine.com/oracle/or-articles/tropashko4

[191] Guozhu Dong, Leonid Libkin, Jianwen Su, and Limsoon Wong, Maintaining Transitive Closure of Graphs in SQL, *International Journal of Information Technology*, 5(1):46--78, October 1999. Available as http://www.comp.nus.edu.sg/~wongls/psZ/dlsw-ijit97-16.ps

[192] MySQL AB, MySQL 5.0 Reference Manual, 2008. Available as http://dev.mysql.com/doc/refman/5.0/en/temporary-table-problems.html

[193] MySQL AB, MySQL Bug System. Bug #10327. Available as http://bugs.mysql.com/bug.php?id=10327

[194] Cuong Do, YouTube Scalability, Google Seattle Conference on Scalability, June 23, 2007. Available as http://video.google.com/videoplay?docid=-6304964351441328559

[195] Red Hat Middleware, LLC., Hibernate Shards - Horizontal Partitioning With Hibernate, Version: 3.0.0.Beta2, 2008. Available as http://www.hibernate.org/hib_docs/shards/reference/en/pdf/hibernate_shard.pdf