A Service Oriented Architecture for Personalized Rich Media Delivery

Sascha Tönnies, Benjamin Köhncke, Patrick Hennig, Wolf-Tilo Balke
L3S Research Center
Appelstrasse 9a
30167 Hannover
{toennies, koehncke, hennig, balke}@l3s.de

Abstract

Multimedia streaming means delivering continuous data to a plethora of client devices. Besides the actual data transport this also needs a high degree of content adaptation respecting the end users' needs given by the form of content preferences, transcoding constraints, and device capabilities. When it comes to content editing (like mixing in subtitles or picture-in-picture composition) relying on third party service providers may be necessary. For improved efficiency all adaptation should be done in a service-oriented way, because a lot of adaptation modules can be reused within different adaptation workflows. In this paper we discuss the extensions of Web service frameworks, and present a first implementation of a service-oriented framework for media streaming and digital item adaptation, concentrating on the technical realization of the services. Our experimental results show the praticality of the actual deployment of service-oriented multimedia frameworks.

1 Introduction

In recent years multimedia content provisioning via the Internet has been dramatically increasing. The resulting multimedia systems typically store media content on a dedicated server. Popular systems, especially in the entertainment area, occurred with the classic video-on-demand (VoD) systems. These static and centralized systems stream their content in a certain data format and encoding.

Over the years the variety of different client devices has been steadily increasing. Thus, the content has to be adapted to the special needs of each individual user, respectively his/her technical device. To allow for such an adaptation, user preferences and terminal capabilities as offered by MPEG-21 DIA are used [10]. A lot of work has already been done in the area of these so-called knowledge-based multimedia adaptation systems. These systems use terminal capabilities and content descriptions to adapt the desired multimedia content. All adaptation processes take place on one single dedicated server. Nevertheless, they also offer individual workflows that are created during a planning phase [8]. Handling the multimedia transformation problem, most approaches use classical 'state space planning problem' and typical AI-based planning algorithms.

Nowadays, there are already efforts to decompose such monolithic systems. The most prominent example is the area of simple media streaming, e.g., for IPTV (Internet Protocol Television). Here, the current trend is to use flexible peer-to-peer (P2P) architectures. There are already several implementations available using different overlay topologies. The most popular system is currently PPLive with more than 50 million registered users. However, all these applications rely on a simple data streaming scenario. When it comes to more complex tasks, like media adaptation P2P systems are unsuitable. This is because complex scenarios make great demands on the system components regarding processing power and reliability. In P2P systems these basic conditions cannot be achieved; P2P systems are unreliable and not every peer is able to fulfill cost expensive media adaptation tasks, e.g. transcoding or scaling.

Our primary goal is to combine the best of both worlds: the individual adaptation of multimedia content as it is done in knowledge-based adaptation systems and the flexible, distributed architecture of P2P systems. The idea of service-oriented DIA has already been discussed in developing knowledge-based multimedia adaptation systems, but has largely been rejected for efficiency reasons (see e.g. [9]). In contrast we will show in the course of this paper that it is indeed possible to adapt multimedia content to the special needs of each individual user efficient on its way through the network using a service-oriented architecture, as recent work proposes (see for instance [13]).

In the Web community the decomposition of complex workflows into smaller independent entities is already a common task for business applications. Web services are the basic technology in service-based architectures to implement service-orientation. However, the current Web ser-

vice specification does not support continuous data as the multimedia domain has to handle. Furthermore, Web services are developed for machine-to-machine communication in the area of business transactions and, therefore, not optimized for real time applications. Nevertheless, instead of building monolithic systems the flexible composition of services into suitable workflows for rich media handling is an important goal for system integration and reusability of components. First approaches were discussed in the multimedia community as a brave new topic on ACM MM 2004 [2].

Indeed, the handling of multimedia content serving a plethora of client devices in a personalized fashion puts great demands on the selection of suitable services and the adaptation of media content [15]. For every delivery process a specific workflow has to be created containing multiple multimedia services. Every individual multimedia service fulfills a specific task (e.g., adaptation) respecting the user's preferences, content semantics, terminal capabilities and Quality of Service (QoS) restrictions [3].

All workflows have to be planned and their execution closely monitored to fulfill the QoS restrictions necessary for real time multimedia applications. This includes, among other issues, the seamless replacements of failing services, or exchanging services on-the-fly reflecting the alteration of the underlying networks parameters. This is especially important in mobile environments where handovers between networks can drastically change the requirements. Moreover, the creation and instantiation of such a service chain needs to solve a multi-objective optimization problem considering all possible services (like discussed in [1]), the currently available content and the author's, user's and client device's constraints on the adaptation.

In the course of this paper we focus on the transport- and control protocol to demonstrate our approachs praticality. We will adapt the Web service approach for the area of multimedia applications and create multimedia services that are subsequently composed to form flexible workflows. Our experiments show that personalization of media streams is possible under real-time constraints in a service-oriented way.

In section 2 we present an overview of existing approaches regarding workflow composition and media streaming. Afterwards, we introduce a use case scenario for our architecture which is further described in section 3. A detailed description of our implementation follows in section 5. A first evaluation of our approach is shown in section 6. A summary concludes the paper.

2 Related Work

In the multimedia community a lot of work has been invested in the area of multimedia adaptation systems. In [9] a knowledge-based adaptation framework is presented that supports dynamic composition and execution of multimedia transformations using existing software libraries. A knowledge-based planner computes a sequence of required adaptation steps based on the available libraries and the user requirements. All transformation steps are fulfilled on one single server. They validated their concept using standard multimedia adaptation libraries and a state-space planning engine. A further development of this work is described in [8], here the focus lies on automatic generation of proper adaptation workflows. Moreover, the authors enhanced their framework by integrating semantic languages (like OWL-S) to ease the planning process. A very similar approach, called CAIN, is presented in [6]. They also build workflows from a series of basic content adaptation tools, but instead of only considering technical capabilities of the client device and resource limitations, user preferences are used in addition to retrieve suitable workflows. The introduced adaptation decision taking algorithm relies on a constraint matching problem: resource limitations versus terminal capabilities. If multiple adaptation strategies are found to match these constraints, the one which matches the most constraints of the user preferences is chosen. Nevertheless, all common systems in the multimedia community are focused on adaptation processes that run on a dedicated server, whereas our idea is to distribute different adaptation steps to different service providers.

Also in the P2P community media streaming has recently received attention. In [17] the authors present DONet, a Data-driven Overlay Network for live media streaming. They devise a smart partner selection algorithm and a low-overhead scheduling algorithm to pull data from multiple partners and discuss the key issues of an Internet-based implementation called CoolStreaming. CoolStreaming is somehow the pioneer in the field of IPTV. Nowadays, IPTV applications have seen a dramatic rise in popularity and, furthermore, have received significant attention from industry and academia, e.g. PPLive or Sopcast. Currently, PPLive is the most popular instance of an IPTV application with more than 50 million subscribers.

Due to its popularity many measurement studies for the PPLive system have been performed. The measured results in [7] show significant play-back time lags between different peers and long startup delays. The startup delay is the time interval from when one channel is selected until actual play-back starts on the screen. It depends on the popularity of the chosen channel and takes for less popular channels up to two minutes. Since a large percentage of participating nodes are behind NATs or firewalls the lengthy startup times and high failure rates are inherent problems in P2P streaming systems (see [12]). The experiments in [16] show that the characteristics of P2P file-sharing overlays depend on the application atop the P2P overlay. Since human users in

P2P file-sharing systems can go away from the client machine while it continuous downloading the content, session lengths are long. The opposite behavior arises in IPTV systems. Here users need to be actively present near the client machine to obtain utility from the application. Therefore, session lengths are shorter and client nodes are impatient, because media play-back should start closely.

However, all P2P applications rely on a simple data streaming scenario without any personalization or adaptation of media content. For such complex tasks P2P systems are unsuitable. In a media adaptation scenario great demands are made on the system regarding processing power and reliability. Since not every peer in a P2P environment is able to fulfill cost expensive media adaptation tasks, e.g. transcoding or scaling, we will concentrate on service-oriented media streaming.

In contrast, streaming in a service-oriented way has been already addressed very early in some papers. In [18] the authors extend SOAP messages by introducing additional attributes inside the SOAP envelope. With these attributes it is possible to support boxcarring and batching of messages. After SOAP has become a standard [5] describes another early Web service based approach which relies on the upcoming standard Direct Internet Message Encapsulation (DIME) from that time. In this architecture the SOAP messages were used for delivering the multimedia content metadata and the multimedia content itself was delivered by using DIME. But DIME has been withdrawn and never made RFC status. Nowadays, SOAP with attachments is quite famous and is used in an architecture described in [11]. Here a new Message Exchange Pattern (MEP) for streaming data is defined and this MEP is than implemented in the SOAP HTTP binding. Using SOAP with attachments a single multimedia data segment can be transmitted as a SOAP message package which can include as many parts as needed.

3 Service-Oriented PUMA Architecture

Figure 1 gives an overview of our Personalized Universal Media Access (PUMA) architecture. The central components of the architecture are the workflow preparation, -validation, -instantiation, and -monitoring. Each of these parts is responsible for a different step in the adaptation process. The first step necessary in our adaptation process is the creation of a suitable workflow for the actual adaptation of the content. Workflows are sequences of adaptation steps, each step corresponding to a certain type – a *role* – of media processing which needs to be performed on the stream data.

Such an adaptation workflow can include, among other roles, steps for transcoding, scaling, framerate adaptation and adding subtitles. Our insights on the workflow selection and interoperability validation process are beyond the scope of this paper and have been published in [4].

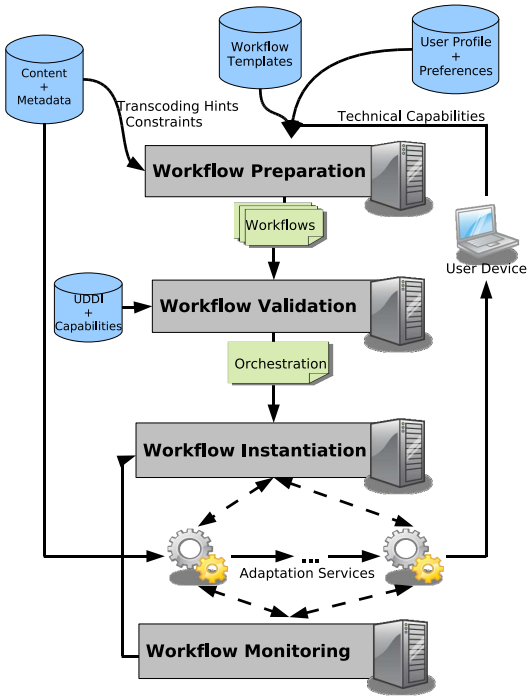

Figure 1. The basic PUMA architecture

Once the service orchestration is prepared and all roles in the workflow have been filled with a suitable service, the next major component in our architecture is taking over. Since in this paper we are interested in the feasibility of service-oriented multimedia applications, also our experiments specifically focus on the workflow instantiation and -monitoring engines. We set up a testbed to investigate all interesting parameters for effective service provisioning.

Service Instantiation Primary task of the workflow instantiation engine is to initialize and start the associated services. Initialization includes establishing the stream connections and the connection to the monitoring service. As we have explicit knowledge about the workflow, we can invoke the services in the correct way and are able to substitute failing services by other service implementations. Since Web services are stateless, we had to introduce a session concept into our architecture. In this way we are able to connect service instances to the workflow and vice versa.

Monitoring of Services The special role of the monitoring service stems from the dynamic properties of streaming over unreliable media, like the Internet or wireless networks. The monitoring of services includes the checking of availability of a service and also the just-in-time replacement of failing services. Other types of adaptation include the replacement of services when the network parameters, e.g. bandwidth or latency, change so that a certain service or workflow is not suitable for adaptation anymore. For the implementation of the monitoring the E^2Mon algorithm [1]

is used in the PUMA framework.

PUMA Protocol To ensure stable communication as well as platform and programming language independency within our framework, we designed and implemented the PUMA Protocol. Since every PUMA framework communication can be assigned to one of two categories, this protocol is split into the PUMA Control Protocol (PCP) and the PUMA Transport Protocol (PTP). The former mainly defines high-level message exchange between Web services. For instance, the login procedure, media search and retrieval, event handling and media stream control messages are covered. Furthermore, WSDL interfaces and message sequences are included in the PCP. Those well-defined interfaces describe exactly how the different Web services can be contacted and controlled, e.g. initialized, started or stopped. PCP relies on SOAP and WSDL. The latter protocol defines low-level data exchange at socket layer and includes specifications of data messages on byte level. Therefore, PTP is based on the well-defined TCP/IP stack and describes how stream socket based connections are established and how data packages have to be formatted. A specification of the PUMA protocol bundle will be available at the project website[1].

4 Use Case Scenario

In recent years, many network providers extended their business model towards provisioning novel value-added services to their customers. Currently phone and television companies are pursuing 'triple play' campaigns, trying to offer three important services: telephony, Internet and television within one network infrastructure. A good example is the Japanese provider NTT DoCoMo who launched the i-mode model very successfully in 1999 [14]. As of 2006 i-mode is reported to have 46.8 million customers in Japan and over 5 million in the rest of the world. The benefit of such a business model is threefold:

- the quality of both services and content can be controlled (e.g., by user feedback) and questionable sources / services can be removed
- QoS can be measured and adequately optimized on a broader scale over the network beyond what individual service providers can achieve
- the close ties between users and network providers, as well as network and content providers allows for simple billing solutions other than micro-payments or complex credit card transfers

Our use case therefore focuses on the role of network providers in a typical entertainment provisioning scenario: media streaming and content adaptation. As a running example we use a personalized Video-on-Demand streaming scenario.

Since most users have similar requirements and network providers are in tight control of their networks, providers will directly provide at least some services such that compatibility and interoperability considerations will not play a big role for simple adaptations. However, by integrating services from other providers added value can also be generated to cater for smaller user groups with special needs (like using translation or subtitling services). In the case of integrating third party providers' services, interoperability is an important issue, so we also address this topic in our multimedia service composition framework [4]. Moreover, the space- and time-dependent nature of media streams poses hard constraints on current service provisioning frameworks.

Example Let us assume a Video-on-Demand service provider who offers popular TV series. Therefore, the video content is purchased from content providers which generally offer high definition media. Since the business model is to serve costumers according to their needs the media has to be personalized.

Imagine a costumer named Mira who wants to watch the newest 'Prison Break' episode on her Laptop. The media content should be in a resolution of 800*480 and encoded in mpeg4 with 25 frames per second (fps). Furthermore, Mira would prefer the media content with a Finnish audio track or at least with Finnish subtitles.

As Finnish is a rarely used language, no content provider may offer such an audio track, but there will be specialized subtitle services available. Thus, Mira's service provider does not have to produce the subtitles on its own, but can obtain them from some third party vendor. To also fulfill Mira's technical requirements a personalized workflow has to be created containing the services. To start streaming the 'Prison Break' episode, Mira contacts her service provider via a Web service call (see figure 2). The service provider's workflow instantiation engine receives this call and instantiates a suitable service chain. The video material is obtained from a content provider and adapted on-the-fly during streaming through TCP-based socket connections. Eventually the adapted material is received by Mira's Laptop and directly played.

5 Implementation

We implemented the service-oriented architecture introduced in section 3 to realize video adaptation on-the-fly during streaming the media content through the network. Web service frameworks offer a set of standards for service-oriented architectures. On the whole we relied on standard

[1] http://www.L3S.de/puma

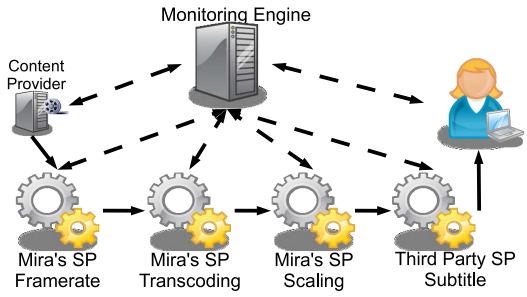

Figure 2. Monitoring of Services: dashed lines represent SOAP messages, compact lines direct TCP based stream connections

components. Since we developed our framework in Java, we chose the Apache CXF Web service framework to ease the Web service creation. Since Web services are stateless and multimedia data is continuous we additionally implemented a session id concept. As this paper focuses on the physical media adaptation and streaming, we reference to [4] for more insights about the Workflow Preparation and Validation. The source code will be available on the project website.

As a first step we developed services to provide multimedia adaptation functionality, since there are no such services on the market. We exploited several concepts to enable a service-oriented media streaming. While searching for a suitable solution we faced the big problem that the multimedia market is a closed source one. Especially Java lacks multimedia support. There is just one official standard API, the Java Media Framework (JMF), on the market which was introduced in 1998. The development of the JMF has been discontinued in 2004. There are some projects relying on JMF, but none of them offers satisfactorily multimedia support.

To solve this problem we implemented a Java wrapper for the command line tools mencoder and ffmpeg. Both rely on the libavcodec codec collection and, therefore, support a lot of codecs and multimedia capabilities.

For delivering multimedia content we implemented the PUMA protocol bundle introduced in section 3.

The instantiation of a suitable service chain for an individual user is done by a monitoring service. This service also monitors the workflow and is responsible for service replacements. The client device only interacts with the monitoring service using Web service calls, as visualized in figure 2. Beside different multimedia adaptation services we also developed the display service which is responsible to show the adapted content on the client side. Furthermore, we developed a content provider which provides a multiplicity of different media files.

As it is usual in service-oriented architectures, we are not bound to a certain programming language. We also integrated a C# .NET Web service in our delivery workflow that perfectly interacts with its adjacent Java services via the PUMA protocol.

In this way we are transferring the media file chunk by chunk via the stream connection. These chunks are then stored in a cache and processed by the adaptation process. The adapted chunks are cached again and transferred to the next service. In case of a service failure, this service is replaced by a similar one immediately and the processing continues with the next required chunk. By storing a number of chunks on each multimedia service provider, we assure that no chunk gets lost. The user does not notice a service failure, because the replacement takes only a few seconds (see [4]).

Thus, we are able to use different services in a chain to change different aspects of the stream and display it on a client device.

6 Evaluation

Server	Processor	RAM
1	4 x Intel Xeon 2.80 GHz	6 GB
2	4 x AMD Opteron 850 2.40 GHz	35 GB
3	4 x Intel Xeon 2.80 GHz	4.4 GB

Table 1. Used servers during the experiments

To evaluate our service-oriented media streaming application we ran several experiments which we describe in this section.

Experimental Setup We performed our experiments using three different servers (see table 1). Since every server consists of four CPUs every server can simulate up to four independent service providers. Therefore, our longest evaluated service chain contains 12 services. Each service provider offers exactly one multimedia service. The workflows are built in a way that after every adaptation step the chunk must be routed through the network between servers. Thereby, we got a realistic service environment, although our servers can act as more than one service provider. We varied the chunk sizes and the number of services included in the workflow during our experiments. Unless otherwise stated, Mira's workflow (see figure 2) is used in our experiments. We used an IBM T60 laptop as play-back device.

The first test measures the scalability of our service-oriented approach. Afterwards, we evaluate the influence of different chunk sizes on startup delay and processing time. At the end of this section we analyze the overall processing and service recovery times.

Server Load As we want to have worst case examinations we need to ascertain the most processing intensive adaptation service. Therefore, we measured the processing time for each type of service, varying chunk sizes and number of concurrently running services. Figure 3 shows the average processing time over 100 runs for a chunk size of 12 seconds (approx. 2 MB) processed on server 2.

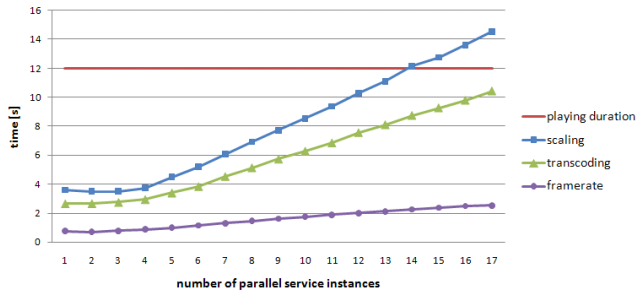

Figure 3. Server 2 Load for 12s Chunks

The results indicate that the scaling service is the most processing intensive service type. As server 2 is a four CPU machine it is obvious that up to four services can run in parallel without an increase of the processing time. Furthermore, the processing time scales well for more than four concurrent services.

If considering Quality of Service aspects service providers must guarantee high reliability. For fulfilling real time constraints the processing time (plus network overhead) must be less than the chunk play-back duration. The results (figure 3) suggest an upper limit of 14 parallel services to provide 100 percent reliability. For an exact estimation of the reliability it is important to consider the variances of the processing times. Table 2 shows the error rate – processing time greater than duration – for 10 to 17 parallel services. For up to 11 parallel services it can be ensured that each chunk is processed in real time.

# Services	10	11	12	13	14	15	16	17
Real Time	100	100	99	90	55	36	20	9
Non-real Time	0	0	1	10	45	64	80	91

Table 2. Success and Error Rates in %

These experiments have reconfirmed the assumption that complex media adaptation tasks make great demands on technical capabilities like processing power. Providing the necessary processing power is feasible for service providers, in contrast to P2P environments. Nowadays, most of the participating devices in current P2P environments do not have enough processing power to adapt media content while also playing the desired media stream.

Arrival rate variance The arrival rate is the time passing between the arrival of two consecutive chunks at the client. For this experiment we do not consider buffering of chunks. Thus, it is important that the arrival time is less than the chunk's playing duration to assure continuous media play-back.

Chunk no.	500 KB	1 MB	2 MB
Startup	4434	7255	14794
02	2388	6070	3880
03	4354	1838	4454
04	4685	1658	5012
05	921	1738	5761
06	1111	1832	
07	954	1807	
08	936	1618	
09	1006	1965	
10	887	1934	
11	1101		
12	969		
13	951		
14	1048		
15	901		
16	904		
17	1040		
18	914		
19	933		
20	952		

Table 3. Average Arrival Rates [ms] of Chunks

Since our test movie was a total of 60 seconds (i.e. 10 MB), we split it up into 20 x 3s (500 KB), 10 x 6s (1 MB) and 5 x 12s (2 MB) long chunks. Table 3 shows the average arrival time (over 100 runs) for each chunk. For the test series with a chunk size of around 500 KB (3 seconds) the results show that the chunks three and four do not arrive in the required time. A continuous media playback would be possible after receiving four chunks resulting in a startup time of around 15 seconds. The startup time for a chunk size of 1 MB (6 seconds) is around seven seconds. The second chunk arrives just in time. Similarly, we measured 15 seconds for the 2 MB (12 second) chunks. In our scenario the most suitable chunk size is six seconds. Therefore, following experiments have been conducted with this chunk size.

Startup Time From the users' point of view a high quality streaming experience requires a fast startup time. To check the quality of our approach we constructed workflows of different length, starting with Mira's workflow. The maximal length of workflows we used was 12 services. To simulate a worst case scenario the additional services added to Mira's original workflow are scaling services. Scaling ser-

vices are the most expensive ones, as seen above.

The startup time has been determined as the time passed from pressing the play button on the display device until the first chunk was completely received.

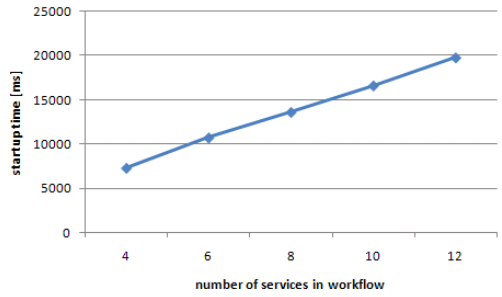

Figure 4. Startup Time for different Workflow Lengths

Figure 4 shows the startup time depending on the workflow length. It is notable that each increase in the workflow length adds a constant delay. A further analysis of the composition of the startup time shows, e.g. for a workflow with four services the startup time of 7255ms (cf. table 3) is composed of 34ms network delay, 146ms IO delay and 7075ms for content processing (see figure 5). Overall, a startup time of around 15 seconds is fast compared to startup times of about 2 minutes as recently reported in P2P streaming applications [7].

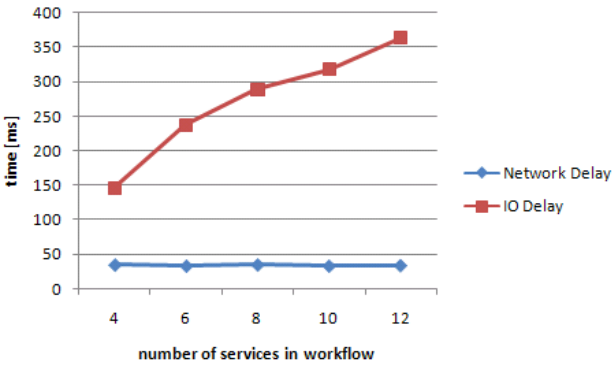

Figure 5. IO and Network Delay for different Workflow Lengths

Knowing that the content processing is responsible for 98% of the delay in this case we conducted another experiment measuring the processing time depending on chunk size and load. Similar to the previous load experiment we used a single four CPU machine running up to 17 parallel scaling processes with varying chunk sizes.

The results depicted in figure 6 show a near linear dependency between chunk size and processing time. The linear increase of processing time is smaller for small chunk sizes, due to the servers more complex threading behavior for bigger chunks. Again we can see that the processing time is greatly influenced by the server load. This renders the application of P2P scenarios, where peers usually have limited processing power, useless for content editing workflows.

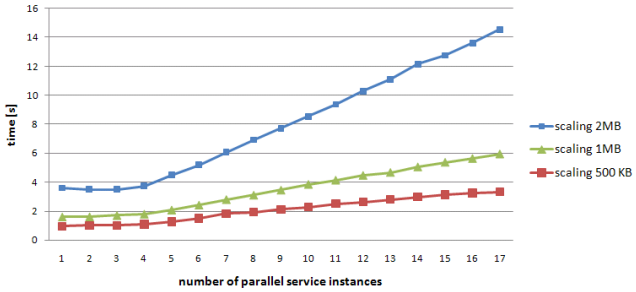

Figure 6. Chunk sizes

Complete Delivery For flawless content delivery it must be ensured that the complete delivery time does not extend the total playing time of the movie. The complete delivery time is the time from the beginning of the play-back until the last chunk is eventually received by the client device. We did experiments using the 1 MB chunks and varying the number of services in the workflow. Figure 7 shows that the time increases by small margins until a workflow length of 12 services is reached.

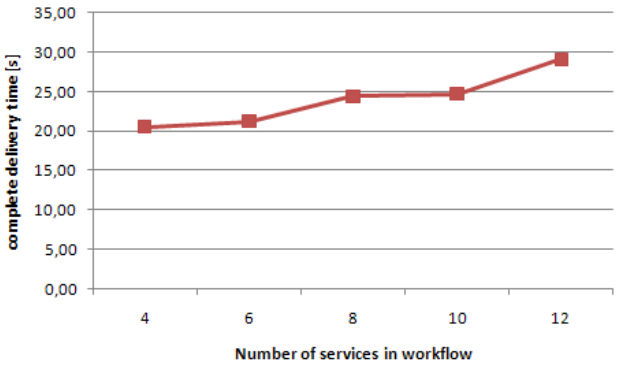

Figure 7. Complete delivery time

The total processing time is much smaller than the playing duration in all workflows under considerations in our experiments.

7 Conclusion & Future Work

In this paper we presented our design and implementation of a service-oriented framework for multimedia content adaptation. Furthermore, we conducted extensive experiments within our framework. We were able to show in contrast to common believe that *content adaptation under real-time constraints is possible in a service-oriented way*. Web services allow the reuse of adaptation components in different personalization workflows. One remaining problem is the lack of support for continuous data streams in current Web service infrastructures. Our framework uses a second protocol layer to support the transport of continuous data between Web services.

Our experiments show that the overhead of network and IO operations is rather low compared to the actual content adaptation process (for instance around 200ms as compared to 7s in a four service workflow). It is doubtful whether in other currently available distributed approaches, namely P2P systems, usual peer devices with limited processing power are able to fulfill the computationally expensive personalization task at all. Still, in contrast to P2P systems with startup times in the range of minutes, our system was able to *personalize and deliver* multimedia content after only a few seconds. The actual startup time for an adaptation workflow, personalizing a 60 second movie using four services is 7 seconds. The overall delivery time the time until the whole content object is available at the client is 20 seconds. Even for more complex workflows, containing more personalization steps, the processing and delivery time scales well (i.e. an additional 3s per task for the most expensive service type).

As a direct effect of implementing the workflows as a pipeline of services, the processing time is largely limited by the performance of the slowest adaptation service in the workflow. For providing a continuous stream at the client side, the timely replacement of failing services is needed. For this we implemented the PUMA E^2Mon monitoring protocol [1]. Our evaluations show that the monitoring service we implemented does not hamper the speed of the adaptation process, but allows to replace failing services on-the-fly. Average times for replacing a failing service, from the detection of the failure until the re-establishment of the content stream, averages to 3.8 seconds.

Also the results presented here are already very promising the problem of service-oriented multimedia applications is more difficult than our testbed shows. Future efforts will go into the development and evaluation of techniques for creation of workflows considering QoS constraints. Furthermore, we want to investigate the use of highly specialized tools for content adaptation to reduce the influence of the processing time as the limiting factor.

References

[1] W.-T. Balke and J. Diederich. A quality- and cost-based selection model for multimedia service composition in mobile environments. In *Proceedings of ICWS*, 2006.

[2] W.-T. Balke and K. Nahrstedt. Multimedia service composition: A brave new topic. Brave New Topics Session within the 12th Annual ACM Multimedia Conference (MM 2004), New York, USA, ACM, 2004.

[3] R. Berbner, M. Spahn, N. Repp, O. Heckmann, and R. Steinmetz. Heuristics for qos-aware web service composition. 2006.

[4] I. Brunkhorst, S. Tönnies, and W.-T. Balke. Multimedia content provisioning using service oriented architectures. In *Proceedings of ICWS*, 2008.

[5] S. Decneut, F. Hendrickx, L. Nachtergaele, and S. V. Assche. Targeting heterogeneous multimedia environments with Web services. In *Proceedings of ICWS*, 2004.

[6] F.Lopez, J. Martinez, and V.Valdez. Multimedia content adaptation within the cain framework via constraints satisfaction and optimization. In *Proceedings of AMR*, 2007.

[7] X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross. A measurement study of a large-scale p2p iptv system. *IEEE Transactions on Multimedia*, 9(8):1672–1687, 2007.

[8] D. Jannach and K. Leopold. Knowledge-based multimedia adaptation for ubiquitous multimedia consumption. *J. Network and Computer Applications*, 30(3):958–982, 2007.

[9] D. Jannach, K. Leopold, and H. Hellwagner. An extensible framework for knowledge-based multimedia adaptation. In *Proceedings of IEA/AIE*, 2004.

[10] B. Köhncke and W.-T. Balke. Preference-driven personalization for flexible digital item adaptation. *Multimedia Systems*, 13(2):119–130, 2007.

[11] G. Lam and D. Rossiter. Streaming multimedia delivery in web services based e-learning platforms. In *Proceedings of ICALT*, 2007.

[12] B. Li, S. Xie, G. Y. Keung, J. Liu, I. Stoica, H. Zhang, and X. Zhang. An empirical study of the coolstreaming+ system. *IEEE Journal on Selected Areas in Communications*, 25(9):1627–1639, 2007.

[13] K. Nahrstedt and W.-T. Balke. Towards building large scale multimedia systems and applications: challenges and status. In *Proceedings of MSC*, 2005.

[14] K. Tsujimura. Docomo's challenge towards new mobile services. In *Proceedings of WWW*, 2005.

[15] A. Vetro. Mpeg-21 digital item adaptation: Enabling universal multimedia access. *IEEE MultiMedia*, 11(1):84–87, 2004.

[16] L. Vu, I. Gupta, J. Liang, and K. Nahrstedt. Measurement of a large-scale overlay for multimedia streaming. In *Proceedings of HPDC*, 2007.

[17] X. Zhang, J. Liu, B. Li, and T.-S. P. Yum. Coolstreaming/donet: a data-driven overlay network for peer-to-peer live media streaming. In *Proceedings of INFOCOM*, 2005.

[18] X. Zhang, D. Towsley, and J. Wileden. Towards interoperable multimedia streaming systems. In *Proceedings of Packet Video Workshop*, 2002.