Future Internet **2011**, *3*, 87-116; doi:10.3390/fi3020087

OPEN ACCESS

future internet

ISSN 1999-5903

www.mdpi.com/journal/futureinternet

Article

A Service Oriented Architecture for Personalized Universal Media Access

Sascha Tönnies *, Benjamin Köhncke, Patrick Hennig, Ingo Brunkhorst **and Wolf-Tilo Balke**

Forschungszentrum L3S, Appelstraße 9a, D-30167 Hannover, Germany;
E-Mails: koehncke@L3S.de (B.K.); hennig@L3S.de (P.H.); brunkhorst@L3S.de (I.B.);
balke@L3S.de (W.-T.B.)

* Author to whom correspondence should be addressed; E-Mail: toennies@L3S.de;
Tel.: +49-511-762-17723; Fax: +49-511-762-17779.

Received: 6 January 2011; in revised form: 3 February 2011 / Accepted: 18 March 3011 / Published: 1 April 2011

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 content preferences, transcoding constraints, and device capabilities. Such adaptations can be performed in many ways, usually on the media server. However, 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 economic reasons this should be done in a service-oriented way, because a lot of adaptation modules can be reused within different adaptation workflows. Although service-oriented architectures have become widely accepted in the Web community, the multimedia environment is still dominated by monolithic systems. The main reason is the insufficient support for working with continuous data: generally the suitability of Web services for handling complex data types and state-full applications is still limited. In this paper we discuss extensions of Web service frameworks, and present a first implementation of a service-oriented framework for media streaming and digital item adaptation. The focus lies on the technical realization of the services. Our experimental results show the practicality of the actual deployment of service-oriented multimedia frameworks.

Keywords: service-oriented architecture; web services interoperability

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 have a low protocol overhead and can, therefore, efficiently broadcast their video content to many different client devices. However, the content is only available in certain data formats and encodings.

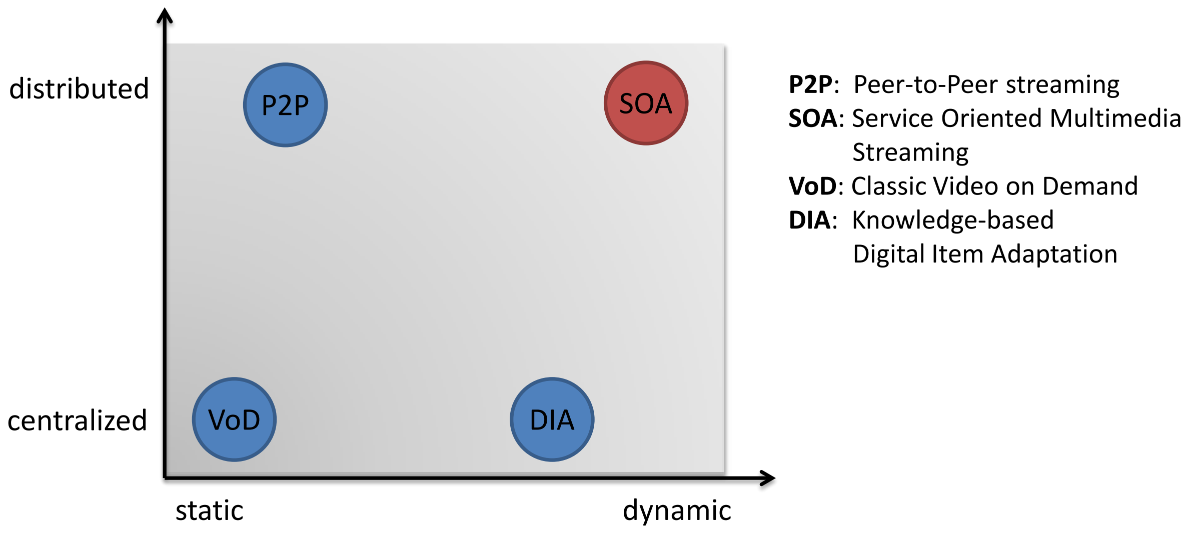
Over the years the variety of different client devices has been steadily increasing. Thus, the content has to be adapted to the special needs of each individual user, respectively his/her technical device. To allow for such an adaptation, user preferences and terminal capabilities as offered by MPEG-21 DIA (Digital Item Adaptation) are used [1]. A lot of work has already been done in the area of so-called knowledge-based multimedia adaptation systems. These systems use terminal capabilities and content descriptions to dynamically 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 [2]. To handle 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 monolithic multimedia systems. The most prominent example is the area of simple media streaming, e.g., for Internet Protocol Television (IPTV). 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 [3] with more than 50 million registered users. Also the European IPTV market is growing fast, for instance Zattoo [4] gained one million users within the last two years. However, all these applications rely on a simple data streaming scenario. When it comes to more complex tasks, like media adaptation, P2P systems are not used. 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 out of the box; P2P systems are unreliable and not every peer is able to fulfill cost expensive media adaptation tasks, e.g., transcoding or scaling, which means that the content is rather static.

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., [5]). In contrast, we will show, in the course of this paper, that it is indeed possible to dynamically adapt multimedia content to the special needs of each individual user in real-time on its way through the network using a service-oriented architecture. Please note, due to the high demands of multimedia streaming, we decided to evaluate our system considering soft real-time constraints.

Figure 1 shows a classification of the different technologies. The shaded areas visualize the efficiency of the systems. Classic VoD systems show a very high efficiency due to their low protocol overhead. With the increase of the protocol overhead due to distribution and dynamics the efficiency decreases.

Figure 1. Technology Overview.


distributed
centralized
static
dynamic
P2P
SOA
VoD
DIA
P2P: Peer-to-Peer streaming
SOA: Service Oriented Multimedia Streaming
VoD: Classic Video on Demand
DIA: Knowledge-based Digital Item Adaptation

We already argued that it is difficult to use P2P technology as bases for adaptation workflows due to the limited devices capabilities. Therefore, we focus on distributing workflow components used in knowledge-based adaptation systems.

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 service specification does not support continuous data such as handled in the multimedia domain. Heinzl *et al.* [6] shows an approach overcoming this limitation by introducing a communication policy that is used in addition to the Web Services Description Language (WSDL) description of a service enabling the implementation of applications with no standard communication requirements, e.g., data streaming. In addition, Heinzl *et al.* [7] introduces an extension of the Web service specification, *i.e.* Flex-SwA, allowing the transport of continuous data over Web services enabling, e.g., streaming. 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 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 [8].

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 [9]. 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 and terminal capabilities.

Example Imagine a user who wants to get a video streamed to a personal digital assistant (PDA) over the Universal Mobile Telecommunications System (UMTS) network. Since, the user has limited free data volume and the PDA imposes some strict technical constraints, e.g., display size and resolution, bandwidth would be wasted if the video is streamed in a resolution higher than the best possible display resolution. Thus, it is reasonable to adapt the video content during delivery to the PDA by creating a specific service chain including a scaling service. On the other hand, a user with a multimedia PC has entirely different technical constraints. A PC has much more processing power than a PDA and usually a high definition monitor. A re-scaling of the media stream may not be needed, but users might want the video with specific subtitles. In that case it is necessary to add the desired subtitles to the media content by integrating a subtitle service into the workflow.

All workflows have to be planned and their execution closely monitored to fulfill the Quality of Service (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 (as discussed in [10]), the currently available content and the author's, user's and client device's, constraints on the adaptation. The actual workflow creation, service discovery and monitoring in our architecture is not the focus of this paper, but can be found in [11].

In the course of this paper we focus on the service chain instantiation to demonstrate our approach's practicality. 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. Furthermore, we will demonstrate that our approach is faster as compared to other distributed streaming systems. In Section 3 we introduce a use case scenario for our architecture which is further described in Section 4. We then present an overview of existing approaches regarding workflow composition and media streaming. A detailed description of our implementation follows in Section 6. A first evaluation of our approach is shown in Section 7. 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 [5] 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. The concept was validated using standard multimedia adaptation libraries and a state-space planning engine. A further development of this work is described in [2]; 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 [12]. Here, also workflows are built 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. In summary, 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 [13] the authors present DONet, a Data-driven Overlay Network for live media streaming. A smart partner selection algorithm and a low-overhead scheduling algorithm to pull data from multiple partners are presented. Moreover, the key issues of an Internet-based implementation are discussed, called CoolStreaming: the pioneer in the field of IPTV (Internet Protocol Television). 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 results in [14] show significant playback time lags between different peers and long startup delays. The startup delay is the time interval from when one channel is selected until actual playback 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 participate nodes are behind network address translation (NAT) boxes or firewalls the lengthy startup times and high failure rates are inherent problems in P2P streaming systems (see [15]). The experiments in [16] show that the characteristics of P2P file-sharing overlays depend on the application atop the P2P overlay.

In typical P2P file sharing systems content is downloaded in the background before consumption by the user, therefore session lengths will be quite long. In contrast, in IPTV systems users are actively present and want to start consuming content right away. Therefore, session lengths are shorter and client nodes are impatient, *i.e.*, the media playback has to start in a timely manner.

Moreover, 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. But not every peer in a P2P environment is able to fulfill cost expensive media adaptation tasks, e.g., transcoding or scaling.

To solve the aforementioned problem, streaming in a service-oriented way has already been addressed very early in some papers. In [13] the authors extend Simple Object Access Protocol (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 had become a standard, [17] described another early Web service based approach which relied on the upcoming standard Direct Internet Message Encapsulation (DIME). In this architecture, the SOAP messages were used for delivering the multimedia content metadata and the multimedia content itself was delivered by using DIME. However, DIME has since been withdrawn and was never given a request for comments (RFC) status. Another efficient method for sending binary data to and from Web services is the Message Transmission Optimization Mechanism (MTOM) [18], implementing XML-binary Optimized Packaging (XOP) for SOAP messages. MTOM uses the multipurpose Internet mail extensions (MIME) to package the message after XOP processing. But nowadays SOAP Messages with Attachments (SwA) are quite popular and are for instance used in the architecture described in [19]. Here a new Message Exchange Pattern (MEP) for streaming data is defined and this MEP is then implemented in the SOAP Hypertext Transfer **P**rotocol **(**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. A current approach introduces a service-oriented infrastructure for multimedia processing using Flex-SwA [7] and is described in [20].

In the Web service business world, although business process execution languages (like e.g., Business Process Execution Language **(**BPEL) [21], Yet Another Workflow Language (YAWL) [22], or Business Process Modeling Notation **(**BPMN) and workflow management systems are closely related. However, the workflows we use in the multimedia domain are not comparable to the business process models, and thus, the available languages are not readily applicable. The main difference to general purpose languages is the inclusion of continuous data streams in our workflows. Only a few extensions to Web service description languages to support continuous data streams exist. Again, most are not yet considered to be ready for use [19,23,24].

Moreover, the workflows currently needed by our architecture are simple compared to the flexibility of the modeling constructs available in the general purpose workflow frameworks. As a result, we use our own Resource Description Framework (RDF)-based workflow model and language with added support for describing the streaming of data between participants.

For the actual instantiation systems already exist, such as IRS-III [25], which can create service compositions using goal based reasoning. However, creating and maintaining the domain ontology needed for the reasoning process is a complex and time-consuming task. As a result, our system works on an extensible set of template workflows for the most common multimedia applications.

In order to describe the capabilities of our multimedia services, OWL-S [26] was chosen on the premise that it is an established standard for semantic Web services. Furthermore, it allows the selection of Web services not only based on their WSDL description [27]. In contrast, OWL-S service descriptions do not only contain information about the invocation of a service, known as service grounding. They also contain information about "what a service does", and "how it does it", known as service profile and service process model, which allows to select services based on capabilities. Such semantic matchmaking techniques have been investigated, e.g., in [28,29].

To our knowledge, the only system to actually validate interoperability between Web services was developed by Foster *et al.* [30]. It uses a translation of choreographies and orchestrations into finite state automata, in this case a labeled transition system. However, this approach does obviously not support continuous data streams.

The checking of interoperability between services is also similar to problems faced in the multi-agent systems domain. Generally, multiple ways exist to check whether interoperability between agents is possible. Recent work especially focuses on checking the conformance of policies with respect to global communication protocols [31]. For the actual conformance tests, different approaches can be used. Popular instances are bisimulation [32] (or a variant of it, see e.g., [33]), as well as the checking of execution traces [34].

In any case, the conversation protocols and policies necessary for checking interoperability can be described using the composite action models as done by various Web service description languages. Common to these description languages is the modeling of actions in form of input, output, precondition, and effects (IOPE) objects. Instances of such languages are OWL-S (an ontology build on top of the Web Ontology Language), Web Services Choreography Description Language (WS-CDL), BPEL, Web Service Modeling Ontology (WSMO) [28], or systems like IRS-III [25].

Respective languages from the multi-agent systems domain are given in [31] with a foundation in Action Logic, for instance systems like Golog [35].

3. Use Case Scenario

Consider the following scenario: Per is a stock broker. While he is on his way to his office he wants to receive the latest news regarding stock markets from abroad on his smartphone. Finally, arriving at his office, he switches to his office desktop PC and continues watching stock market news.

Per's smartphone, as well as the desktop PC, run a sophisticated application automatically selecting the best from a list of available information sources based on Per's content preferences and technical capabilities. This basic task of the application can be accomplished in different ways, so the application designer may have envisioned several possible service chains with different workflows.

Detailed Example: Another Monday morning and Per has to go to work. He switches on his Android smartphone application which offers several settings of how to react to different events, e.g., network change or low battery event. Starting the application at home he receives the latest stock market news and due to the high battery load of his smartphone and the available WLAN he retrieves them via a video stream. Since watching videos require a lot of energy his smartphone's battery load gets lower and lower. While riding the train the battery load reaches 20%. The application automatically switches into audio mode to save battery capacity. Arriving at the train station only 10% of the battery load is left and Per retrieves the latest news via instant messaging. Finally, Per arrives at his office, starts his Desktop PC and runs the application there. Now the content is streamed in high definition resolution as a picture in picture stream combining several information providers.

Instead of building complex multimedia applications (like a monolithic Video-on-Demand based information system), providers need the flexibility to dynamically adapt to a variety of content requests and support a plethora of (mobile) client devices. Service providers need to acquire the content from several content providers that will provide the content and its description in a high-quality format suitable for different adaptation steps, like scaling or re-encoding. Then the content has to be adapted to the end user needs and the respective device's technical profile.

Though the basic workflows are generally well-understood and can be used in the form of templates, the digital item adaptation (DIA) can usually be performed in several ways. To dynamically realize a best possible instantiation user preferences, transcoding hints, and technical capabilities (e.g., given by MPEG-7/21) have to be considered. This complex optimization problem of workflow selection has already been addressed in recent literature; see [36] or [1]. Moreover, to allow for optimizing the necessary adaptations of the content on its path through the network, as well as being able to flexibly choose and instantiate workflows, recent work proposes that the actual adaptation should be done in a service-oriented way (see for instance [37]). The respective techniques strongly differ from the one-size-fits-all that is still prevalent in most multimedia delivery scenarios. Providers have to use service components (like formalized by Web service standards) and compose them into the final workflow.

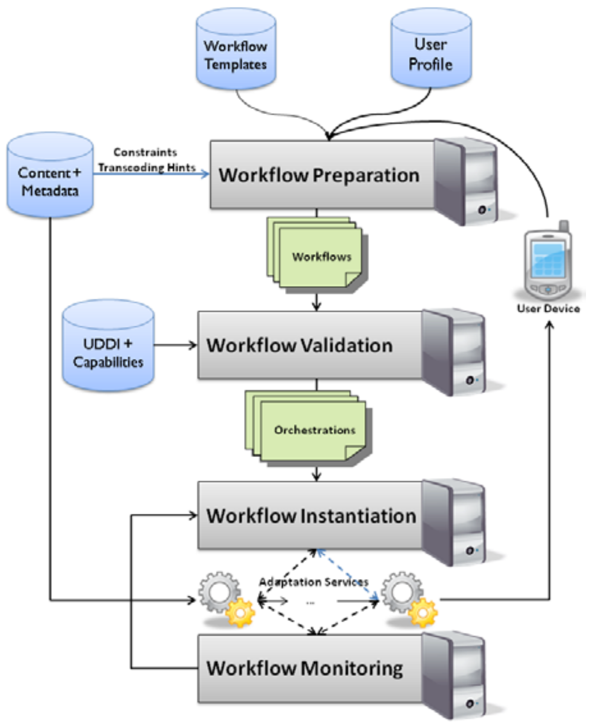
Since most users have similar requirements and network providers are in tight control of their networks, providers will directly provide at least some services such that compatibility and interoperability considerations will not play a big role for simple adaptations. However, by integrating services from other providers added value can also be generated to cater for smaller user groups with special needs (like 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 [9]. Moreover, the space- and time-dependent nature of media streams poses hard constraints on current service provisioning frameworks.

In the following section we will in detail discuss our Personalized Universal Media Access (PUMA) architecture that allows flexibly composing, instantiate and monitoring multimedia services for media streaming.

4. Basic PUMA Architecture

Figure 2 gives an overview of our service-oriented 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.

Figure 2. Basic Personalized Universal Media Access (PUMA) architecture.

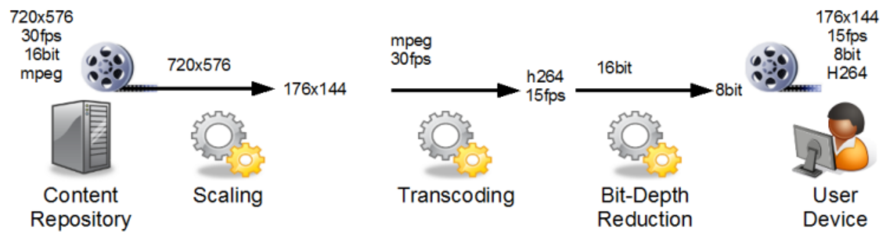


4.1. Workflow Preparation

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 (as illustrated in Figure 3). Such an adaptation workflow can include, among other roles, steps for transcoding, scaling, framerate adaptation and adding subtitles. Existing approaches based on goal-based reasoning utilize a full domain ontology of the roles, capabilities and attributes required in the adaptation process. Hence, they try to create a suitable service orchestration directly using a reasoning process to compute the necessary step for reaching a certain goal. One example of such a system is KMi's IRS-III 25.

Figure 3. Example adaptation workflow for a mobile device.



However, highly dynamic compositions with QoS-based models are not supported by the available reasoning engines for Web service composition. Currently there are first proposals to extend BPEL to support such functionality, however they lack implementation. Therefore we take a different approach for constructing our workflows, although still being based to a certain degree on reasoning within domain ontologies. We assume that specific workflows are needed for the different types of adaptation processes, depending on the available content and other variables. Such workflows can be engineered by multimedia specialists and specifically tailored towards the needs of the required adaptation. The selection of which workflow is most suitable is based on rules specified by domain experts, but is influenced by other criteria, such as projected service costs.

Digital Item Selection

Many different factors need to be taken into account for selecting, creating and modifying the workflows to suit a specific adaptation goal [1].

- *User Preferences* play a major role in selecting the content, as they tend to closer specify the user's interest in terms of topics or, in this case, actor, genre or director. However, such preferences can be used to also decide which content adaptation is more important for the viewer, e.g., picture size over bandwidth utilization, subsequently influencing the type of workflow which is selected.
- The user *device's capabilities* help directly defining, among other things, the output stream's resolution, bit-depth, frames per second and encoding format. In one instance this is done by directly providing the properties of a device, like display size. Other than that, device's capabilities can be seen as e.g., a function of available processing power and bandwidth, for selecting a suitable codec.
- On the other side, *Transcoding Hints* describe what adaptation the author of a content object deems as possible for still being within useful limits, e.g., scaling a scene down to a factor 1:4 would be within limits for the author, to get his point across, but smaller scaling or cropping definitely would not be.

4.2. Workflow Validation

Once the workflow is generated, the roles in it need to be filled with matching services, which will be discovered by the traditional approach of using a central Web service registry. However, since WSDL is not capable of describing Web services expressively enough for matching the services capabilities and behavior with the roles in our workflows, we utilize a semantically rich description language, namely OWL-S, which allows to specify capabilities by referring to our service ontology and process models to specify the services communicative behavior.

The workflows used in our approach resemble choreographies, as they are specifying the interactions between the adaptation services and other parts of our architecture. These interactions are required for stream control, such as starting and stopping the playback. Similar to approaches from multi-agent systems, we use a form of bisimulation [33] to ensure, that the messages exchanged between participants are legal w.r.t a globally defined protocol. A more detailed description of the verification step is given in [11].

Quality of Service Issues

Another aspect for selecting the right services for the roles in a workflow are the cost associated with each service. These costs do not just consider the execution or failure costs for that service but also a variety of costs depending on device capabilities or the execution environment like e.g., QoS parameters. The approach in [38] proposes an extensible framework for modeling such costs for individual applications. For instance, it is possible to make dynamic bandwidth measurements for selecting providers with highest throughput. Suitable end-to-end measurements are possible e.g., with Self Loading Periodic Streams (SLoPS) [39]. In this case a provider with a low throughput would be assigned higher costs. All the costs can then be aggregated in a cost function which will also be used for the service selection task. The deployment of a complex cost function is very application dependent and therefore out of the scope of this article, but definitely has to be expected.

4.3. Workflow Instantiation and Monitoring

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

4.3.1. Service Instantiation

Primary task of the execution 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*.

4.3.2. 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. Monitoring 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 any more. For the implementation of the monitoring, the E^{2}Mon algorithm [10] is used in the PUMA framework.

5. A New Approach for Selection and Validation of Workflows

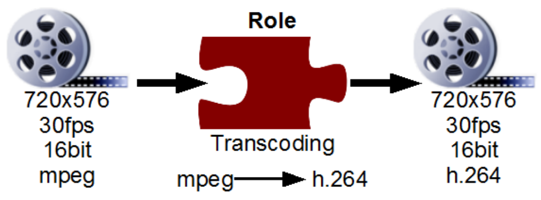
In our architecture a service composition is built in three steps. It starts with the selection of a suitable multimedia workflow, and then discovers services that fit into the workflow, and finally validates the interoperability between the services.

5.1. Workflows

The starting point for all our service compositions is a ranked list of multimedia workflows. Different from general purpose workflow management systems, our workflows are very specific and only require a small subset of the possibilities of general purpose workflow languages.

Consider our example of media streaming. It consists of fetching, adapting and delivering the content data to the user. Each step, or role (see Figure 4) manipulates a certain aspect of the content stream, e.g., it changes the encoding of the stream from the content providers high quality mpeg format to a low bandwidth codec used by mobile phones, like h.264.

Figure 4. Role example: Transcoding a stream from one format to another.



In our architecture, the multimedia workflows are represented using a straightforward action-like model, consisting of roles and data objects. Data objects model the changing of properties of multimedia content after each step. During the design process we considered using a reasoning system to derive a workflow from a set of attributes of the source content data and the given requested target format. Since we are facing real-time constraints, a less computational heavy approach was chosen. For practical reasons we decided to create a set of suitable template workflows for those multimedia processes in our scenarios, which will then be selected based on the tasks required, e.g., if the source and destination formats differ in size and encoding, a workflow template reflecting the necessary transformation will be selected. Still, a simple domain ontology is needed to define the types of adaptations and allow enough reasoning about service capabilities and interoperability, as well as the initial workflow selection.

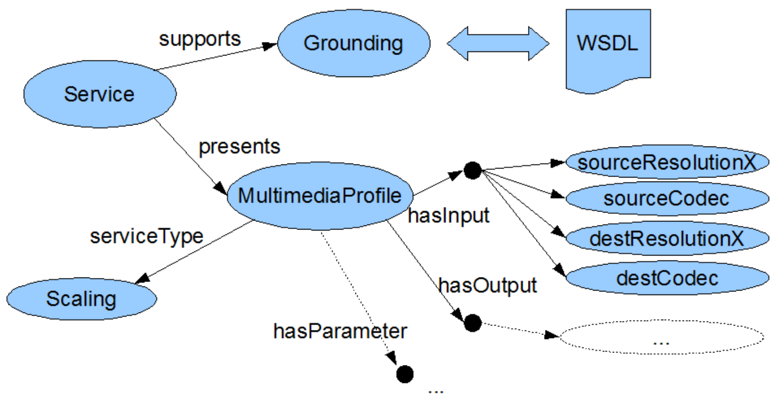
Workflows can have different goals, e.g., can be optimized for quality, network efficiency or costs. As a result the system ends up with a ranked list of suitable workflows.

Figure 3 shows an example adaptation workflow with steps for scaling, transcoding and bit-depth reduction to adapt the source content object for the user's display device. Starting on the left with a high resolution source stream, each service processes the data. Our sample workflow starts with a scaling role to reduce the size. The transcoding step decodes the stream and re-encodes it using a different codec, while the last step is a service specialized in adapting the color information for the limited capabilities of the user's display device.

5.2. Selection

The second step deals with finding suitable services to fill in the roles in the adaptation workflow. For this, first a semantically rich description of each service's capabilities is needed. OWL-S in combination with our domain ontology is used to describe the services we plan to use in our adaptation process. Finding candidate services is based on the OWL-S description of the services, more specifically on the Service Profiles ("what a service does" [26]). Figure 5 shows a part of the OWL-S description of an available scaling service.

Figure 5. OWL-S Description of a scaling service (profile sub-graph).



Each of the roles in the workflow contains information about the required input and output parameter types and values of a manipulation step. Using the OWL-S service profiles, it is then possible to select candidate services that suit the necessary roles.

The result of the selection process is a list of services for each role in the workflow. This is repeated for every workflow in the list until all roles are filled with services.

5.3. Validation

In a monolithic system where all components implemented w.r.t. to the same implementation specification, validating the interoperability of the components is not a big issue. However, we assume that existing services are offered by different third-party providers and are being reused by different systems. It is important that services can interoperate with each other, not only on the level of messages (invocations), but also on the level of behavior (interactions) according to complex protocols.

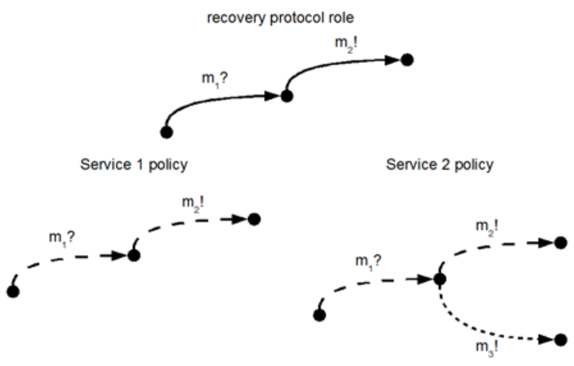
While multimedia workflows often resemble a pipeline (e.g., of consecutive adaptation steps), the continuous nature of the data stream requires a high amount of protocol messages for stream flow control, monitoring and recovery after failures. Especially for detecting and replacing failing services the correct behavior of services prior or later in the chain is essential.

For describing these capabilities of a service that lies beyond the observable behavior (as provided in the WSDL and OWL-S descriptions) additions to the Web service description languages are needed, e.g., WS-CDL+C [35]. In multi agent systems [31], there are a plethora of approaches to verify whether a set of agents can produce a conversation. By using conversation policies and conversation protocols, it is possible to describe the requested and actual behavior of a service.

For multimedia workflows, a service is interoperable with other services in the workflow, if the message it is expecting to receive or sending is in line with the expected behavior of the role it supports. In the following sections a part of the monitoring protocol is used as an example to explain the interoperability check. The process of reestablishing the data connection to a replacement service, using a handshake message as a reaction to the “Recovery” event triggered by the Monitoring Service, is such an expected behavior for certain roles in the workflow.

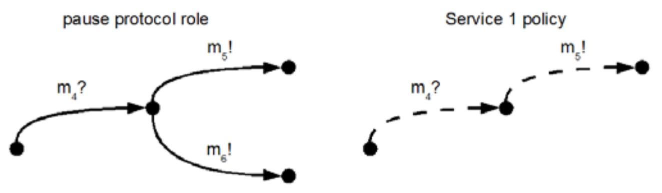
Figure 6 gives an example for one of the service interactions for recovering from a failed service, m? represents incoming messages, m! outgoing messages.

Figure 6. Recovery: protocol specification and service policies.



In the first example, the protocol specifies that the receiving of message m_1? (*recovery after failure*) has to be followed by sending m_2! (*open connection*) to another participant. In the example, the policy of service 1 is conformant w.r.t. the requested protocol. This is different for service 2, which additionally can send a message m_3! (*continue processing*) that is not supported by the protocol in this case.

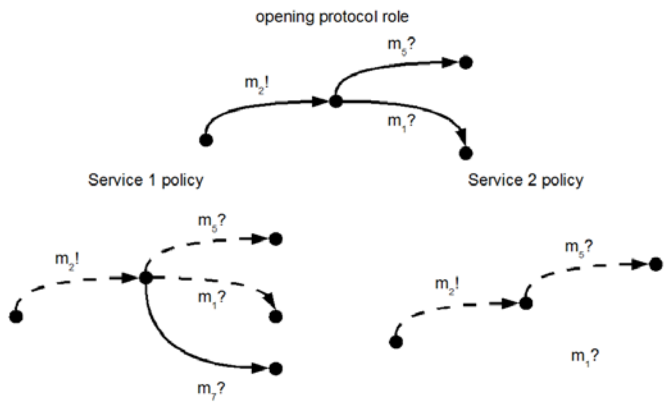
Figure 7. Stop: protocol specification and service policies.



For a different part of the protocol, to temporary pause a stream (see Figure 7), the interaction requires the reception of stop request m_4? to be followed by sending message m_5! (*disconnect*) or message m_6! (*pause processing*). Service 1, which does not support the method using m_6!. However, service 1 is interoperable in this case, since all the possible messages it can send are included in the protocol. Protocol specifications also include the expected interactions following an outgoing message, as shown in Figure 8.

For this case, the protocol requires that opening the connection via message m_2! is to be followed by waiting for disconnect requests m_5? or recovery messages m_1?. Service 2 does not support recovery (via m_1?), and as such is not usable in the composition. Service 1 differs, as it additionally listens for a diagnostics message m_7?, which is not required by the protocol and so does not invalidate interoperability.

Figure 8. Opening: Protocol specification and service policies.



6. Streaming Protocol for Service-Oriented architectures

To ensure stable communication as well as platform and programming language independency within the PUMA framework, we designed and implemented the PUMA Protocol. It defines the communication modality between participating Web services and covers the following use cases.

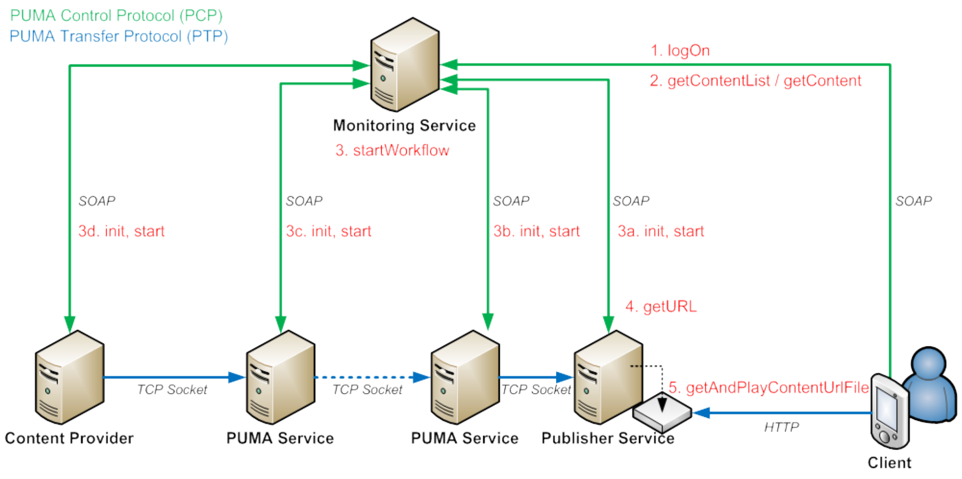
1. User starts client, logs on to the PUMA system and searches for a specific resource.
2. User queries for some content according to his profile.
3. User selects out of the available content.
4. User receives content.

Since every PUMA framework communication can be assigned to one of two categories, our 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 PUMA Control Protocol. 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.

Figure 9 demonstrates the process from login until the client can get and play some media content. It also visualizes the cooperation of both protocols.

To cover the set of use cases, we present the typical procedure from the client's point of view in the following. Mainly the client to Monitoring Service communication consists of three phases: login, media search / retrieval and event handling.

Figure 9. PUMA content delivery sequence.



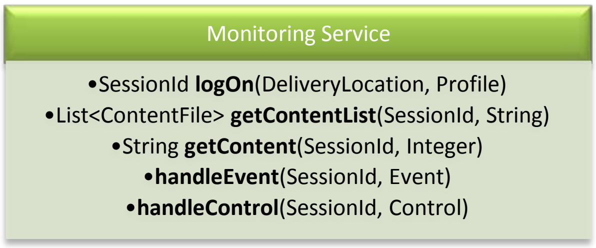
Thus, in more detail, the client first connects to the Monitoring Service providing its DeliveryLocation and Profile bean, both being JavaBeans. A Session ID is generated, allocated and returned to this client as response. Next, the client either searches for content or directly initiates a workflow to get a specific file. After the first processed chunk arrived at the Publisher service, its HTTP address is delivered to the client via Monitoring Service so the client can start to play the media stream. While processing, it is possible that any events arise, e.g., the battery status of the user's device gets low or the network status changes. In this case a method at the Monitoring Service is called while providing an event bean, which includes all information about the occurred event. After receiving and processing the event, the Monitoring Service reacts properly to it. Notice, that events can also be triggered by all services except the Monitoring Service.

6.1. PUMA Control Protocol (PCP)

The Monitoring Service interface is intended to cover client login, search operations, handle events and playback control operations.

Figure 10 shows an overview of this interface.

Figure 10. Monitoring Service interface overview.



Monitoring Service

- SessionId **logOn**(DeliveryLocation, Profile)
- List<ContentFile> **getContentList**(SessionId, String)
- String **getContent**(SessionId, Integer)
- **handleEvent**(SessionId, Event)
- **handleControl**(SessionId, Control)

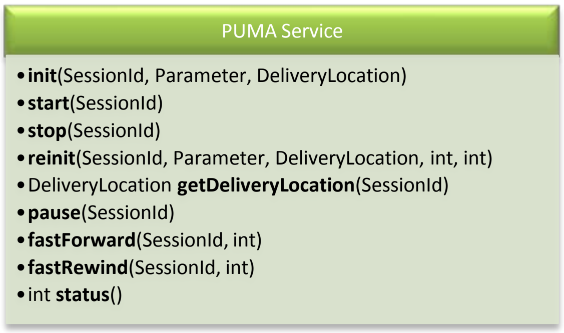
For each communication information unit, *i.e.*, DeliveryLocation, Profile, Event and Control, we defined a complex XML data type (see [40]). In this section, we will shortly describe them.

A *DeliveryLocation* contains information about the network location and the listening port for incoming data transfer connections of a service. A *Profile* is composed of a user profile holding user preferences and a device profile containing device constraints and capabilities. The attributes in *ProfileUser* and *ProfileDevice* are meant as a basis for personalization and can be easily extended. An *Event* encapsulates an event and all necessary information to handle it. A *Control* encapsulates all playback associated information and is used to control playback at client side.

Next, the Web service interfaces of the PUMA framework are depicted. We defined two interfaces: the Monitoring Service interface and the PUMA Service interface. For further insight to the WSDL code, please visit the PUMA project website.

The PUMA service interface (see Figure 11) is meant as a basis for all other PUMA framework Web services. It covers chunk processing and playback operation changes like starting and stopping a stream, redirecting input ports/output locations, status notifications.

Figure 11. PUMA Service interface overview.



PUMA Service

- **init**(SessionId, Parameter, DeliveryLocation)
- **start**(SessionId)
- **stop**(SessionId)
- **reinit**(SessionId, Parameter, DeliveryLocation, int, int)
- DeliveryLocation **getDeliveryLocation**(SessionId)
- **pause**(SessionId)
- **fastForward**(SessionId, int)
- **fastRewind**(SessionId, int)
- int **status**()

6.2. PUMA Transport Protocol (PTP)

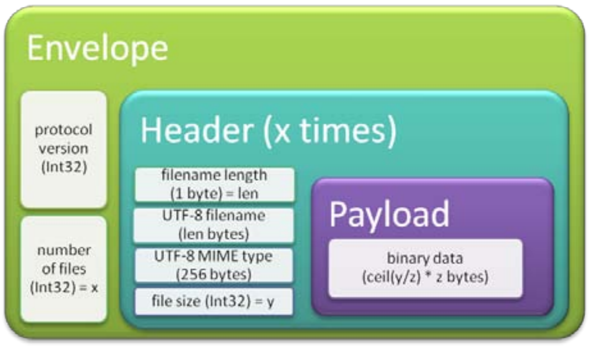
In the following the PUMA Transport Protocol and its technical specification is described. If a transmission of media files, respectively chunks, should be performed between the Web services in the workflow, all files are sent using this protocol. The PTP consists of three parts: Envelope, Header and Payload. Figure 12 depicts the format of PTP.
Envelope

The envelope covers the whole transmission and is only transmitted once. It contains the protocol version, which is an Int32. Int32 must always be 4 bytes big endian (high byte first) encoded. The second part of the envelope is another Int32 value, defining the number of files to be transmitted. This value is intended for a concluding verification if the amount of transmitted files matches the expectation.

6.2.1. Header

The next part of the protocol is the header. The header is transmitted *x* times together with the payload, where x is equals to the number of files defined in the envelope. The first field of the header is the filename length of each file to be transmitted. This information is 1 byte describing the number of characters of the filename, which will be expected and transmitted within the next field. The filename must be UTF-8 encoded to guarantee platform interoperability, e.g., between Java and C#. Next, the MIME type is being transmitted. The reserved space for this information is 256 bytes. Finally, the header contains the size of the file (*y*), which is an Int32 value, respectively 4 bytes big endian. If the filename contains the word "FINISHED", the sending service signals that the last file was transmitted and the transmission ends. This way the expected number of files in the envelope can be checked for correctness.

Figure 12. PUMA Transport Protocol overview.


Envelope
protocol version (Int32)
number of files (Int32) = x
Header (x times)
filename length (1 byte) = len
UTF-8 filename (len bytes)
UTF-8 MIME type (256 bytes)
file size (Int32) = y
Payload
binary data (ceil(y/z) * z bytes)

6.2.2. Payload

With the file size given in the header, the payload just contains accordingly that much bytes of file data. This binary data is distributed on packet with the size of the buffer (*z*). Hence, we transmit ceil(*y*/*z*) * z bytes within the payload for every file, whereas the last packet can be smaller.

6.3. Requirement for Third-Party Providers

Third party providers who want to implement services for our system have to implement all functions described in the WSDL file provided on the PUMA website [40]. An overview of the functions can be seen in Figure 11. In addition both PUMA protocols, *i.e.*, PCP and PTP, have to be implemented.

7. Implementation and Evaluation

We implemented the service-oriented architecture introduced in Section 4 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 components. Since we implemented 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 implement a session id concept.

7.1. Service Selection

Given the workflow, the first step in creating the service composition is the selection of suitable services from the service registry. As stated before, we need OWL-S in addition to WSDL for describing our services. Our service registry is realized using the open-source RDF store Sesame [41], configured to use a MySQL database to store the RDF graph.

For each of the roles in the workflow, a set of candidate services need to be discovered and selected from the set of available services. This filtering is based on the type (e.g., transcoding) and the available operation of each service.

If not reported otherwise, all measurements are averages over a hundred independent runs and were performed an IBM T43 with 1.5 GB of memory and a 2.0 GHz Intel PentiumM processor.

Figure 13. Selecting service candidates from the repository.

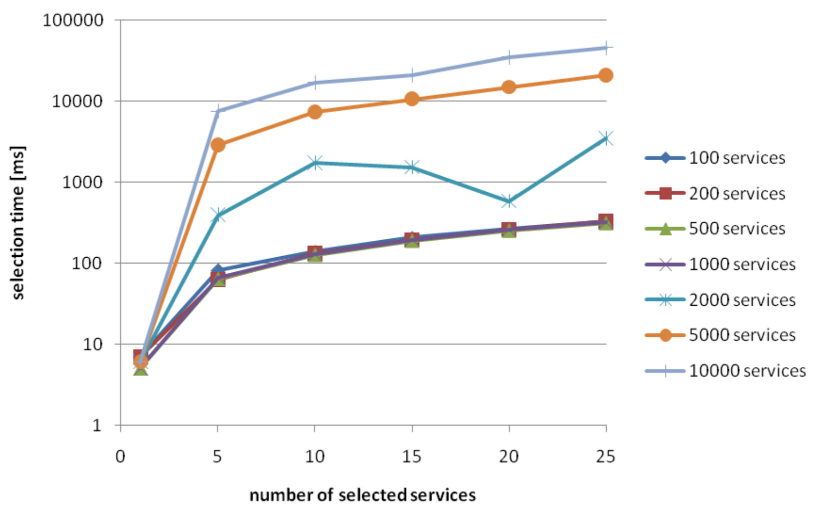


Figure 13 shows the overall time necessary for accessing suitable services inside the service repository. This time is composed of locating the service in the database and retrieving the respective service description. The overall times are reasonably small for repositories containing up to 5000 services, and selecting up to 5 candidates for each role in the workflow. This leads to an offset delay of around 3 s which even in video on demand frameworks is considered practical. A more detailed analysis has shown that locating the service candidates is actually very fast, even for large databases (average <200 ms). However, time for retrieving the service description from the sesame database considerably increases with the number of services. Therefore, selecting new services on-the-fly for replacing failing services is not an option in the current implementation, and has to rely on selecting several services at the initial selection step.

7.2. Workflow Validation

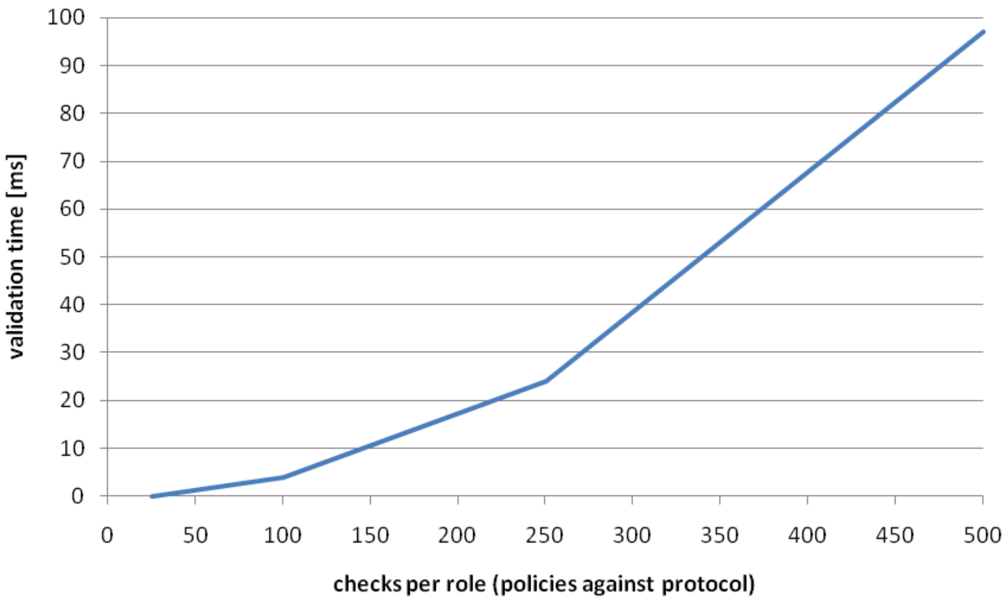
For modeling Web service choreographies and compositions, a number of description languages exist, like Web Service Modeling Ontology (WSMO) and Business Process Execution Language (BPEL), which include the means to define the required message protocols. Using the existing IOPE (Inputs, Outputs, Preconditions and Effects) annotation together with composition process elements, like Sequence, Split, Any-Order and Choice, it is possible to represent the events as described in Section 5 in many of these languages. At the current stage, we specify protocol roles already at the workflow level, using elements from WSCDL [35] for simplicity.

Describing the policy, *i.e.* the behavior of a service, is somewhat more difficult. Though OWL-S basically supports the IOPE annotation of process models the sending and receiving of messages only corresponds to the invocation of individual Web service operations. For the sending of messages to other services, an unbound process is needed, as there is no grounding for the receiving service available. In its current state, checking interoperability is done according to the description in [33], which is a variant of bi-simulation [32].

The protocols we currently use are still simple, so a Java based approach was chosen to check compliance of a policy against the protocol role. When protocols get more complicated and interactions between services increase, transformation of the protocols and policies into finite state automata (FSA) might be needed and checking requires specialized tools, e.g. Spin [42].

Figure 14 shows the time needed for validating the policies of a service against the protocol specification of a role. As we can see, the validation time stays in the area of hundred milliseconds. However, the policies which are stored as part of the OWL-S service description need to be retrieved from the database and transformed into Java objects, which takes considerably longer then the check itself (around 500–700 ms). Note that this materialization can already be performed for several selected services at an earlier step, e.g., during service selection.

Figure 14. Validating service policies against the protocol role.



7.3. Adaptation Services

Each of our services has a well-defined Web service interface which we use for protocol issues (cf. Section 6). Through this interface we can start, stop and monitor the service.

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 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 on the market: the JavaMedia Framework (JMF), which was introduced in 1998. However, the development of the JMF has been discontinued in 2004. There are some projects relying on JMF, e.g., FFMPEG-Java (FMJ) and Jffmpeg, but for us none of them offers satisfactory 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. We encapsulated this wrapper by implementing the PUMA protocol bundle introduced in Section 6. Therefore, as is usual in service-oriented architectures, we are not bound to a certain programming language and are able to deliver multimedia content via different service implementations. For example, we integrated C# .NET Web services in our delivery workflow that perfectly interact with its adjacent Java services via the PUMA protocol.

7.3.1. Experimental Setup

We performed our experiments using three similar servers as video scaling web services (see Table 1). Since every server consists of four CPUs every server can simulate up to four independent PUMA service providers. Therefore, our longest evaluated service chain contains 12 scaling services. 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. All servers are connected via 1 GBit/s connections, except the client which is connected by 100 MBit/s. We used an IBM T41p laptop as client playback device.

Table 1. Servers used during the experiments.

Server	*Service*	*Processor*	*Available RAM*
1, 2, 3	scaling	4 × Intel Core2 Quad 2,67 GHz	4,0 GB
4	content & monitoring	4 × Intel Xeon 2,8 GHz	5,7 GB
5	publishing	4 × AMD Opteron 850 2,4 GHz	33,2 GB
6	Client	1 × Intel Pentium M 1,7 GHz	1,5 GB

7.3.2. 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 15 shows the average processing time over 100 runs for a chunk size of 12 s (approx. 2 MB) processed at server 2.

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

When 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 playback duration. The results (Figure 15) 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.

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.

Figure 15. Server load for 12 s chunks.

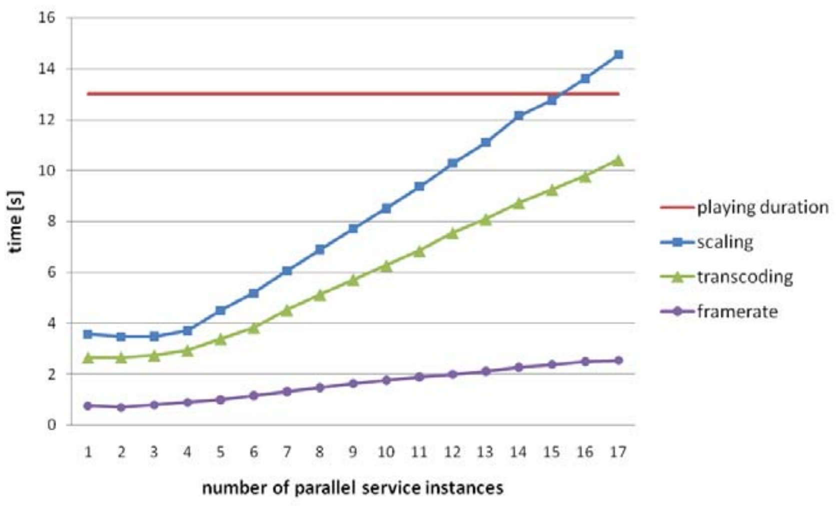


Table 2. Success and error rates in %.

# 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

7.4. Multimedia Workflows

The stream data is not sent through SOAP messages; therefore, we used a separate socket based channel (cf. Figure 9). This means that each service instance has a direct Transmission Control Protocol (TCP) connection to his predecessor and successor. In both ways we are able to provide several services like scaling and transcoding. We also developed a 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, and a publisher service that is responsible for delivering the processed multimedia data via HTTP to the client.

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 9. 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.

7.4.1. 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. As stated before, the maximal length of workflows we used was 12 services. To simulate a worst case scenario all services are scaling services. Scaling services 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 at the publisher service. The pull concept of our PUMA framework, *i.e.*, the client getting the adapted media file stream from the publisher service, is providing an additional average delay not greater than 190 ms for all chunk sizes.

Figure 16. Startup time at client.

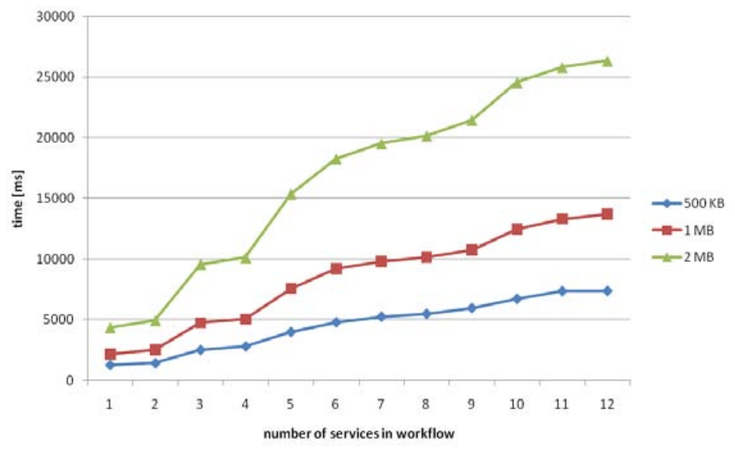
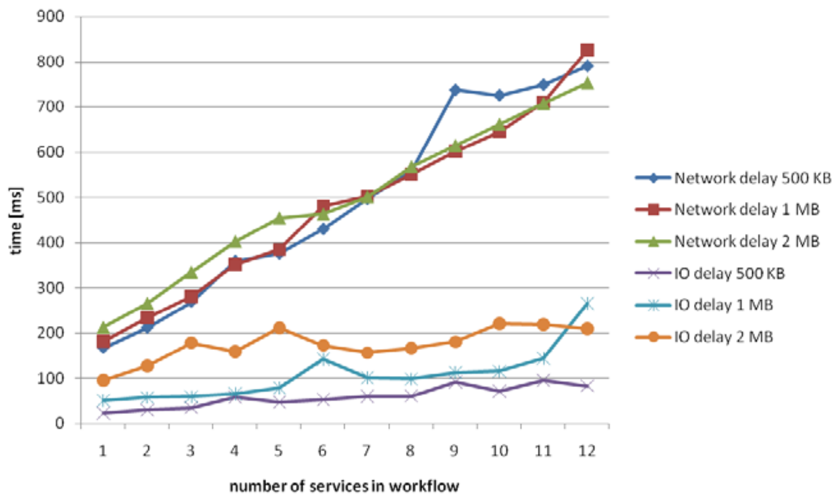


Figure 16 shows the startup time depending on the workflow length. It is notable that each increase in the workflow length adds a constant delay. A further analysis of the composition of the startup time shows, e.g., for a workflow with one service, the startup time of 1995.28 ms with 1 MB chunks is composed of 181.01 ms network delay, 50.66 ms Input/Output (IO) delay and 1763.61 ms for content processing (see Figure 17).

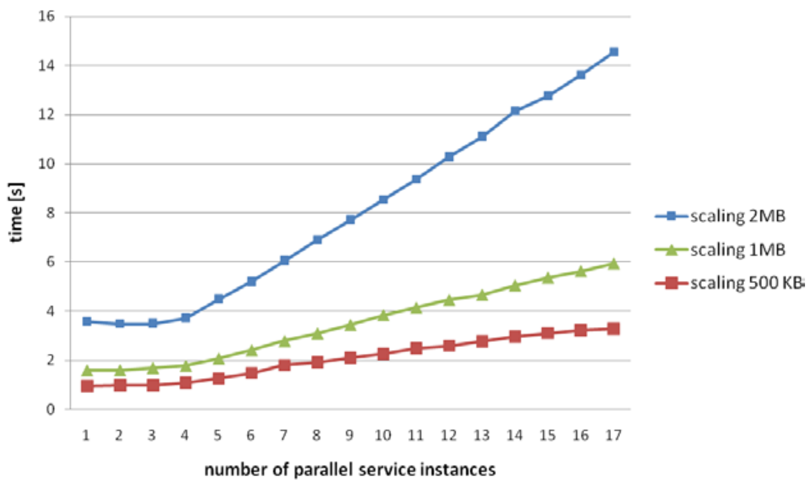
Figure 17. Input/Output and network delay.



Overall, a startup time of around 4 s for 2 MB chunks is remarkably fast compared to startup times of about two minutes as recently reported in P2P streaming applications [14].

Knowing that the content processing is responsible for 88% 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.

Figure 18. Chunk sizes und processing time.



The results depicted in Figure 18 show a nearly 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.

7.4.2. 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 playback until the last chunk is eventually received by the client device. We did experiments using 500 KB, 1 MB and 2 MB chunks while varying the number of services in the workflow.

Figure 19. Complete delivery time and variation range.

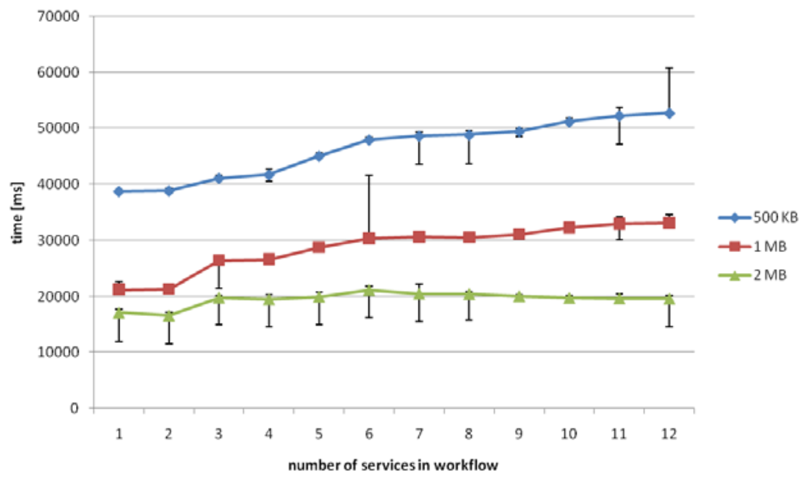


Figure 19 shows that with 500 KB and 1 MB chunks the time increases by small margins until a workflow length of 12 services is reached, whereas the 2 MB chunks need a nearly constant delivery time for each workflow length. The reason for this seems to be a smaller network overhead because of fewer files to transmit.

In most cases the total processing time is much smaller than the playing duration in all workflows under considerations in our experiments. Additionally, one can see the deviation of the average of 100 measures in each case. As we can see, for the 500 KB chunks and 12 services in the workflow a time bigger than 60s is reached. But further examinations of this observations show that this occurred just in 1% of our measurements, so we can consider this as an outlier. This leads to the conclusion that a flawless playback can be expected for all examined workflow lengths.

7.4.3. Service Monitoring

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 previously visualized in Figure 9. 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.

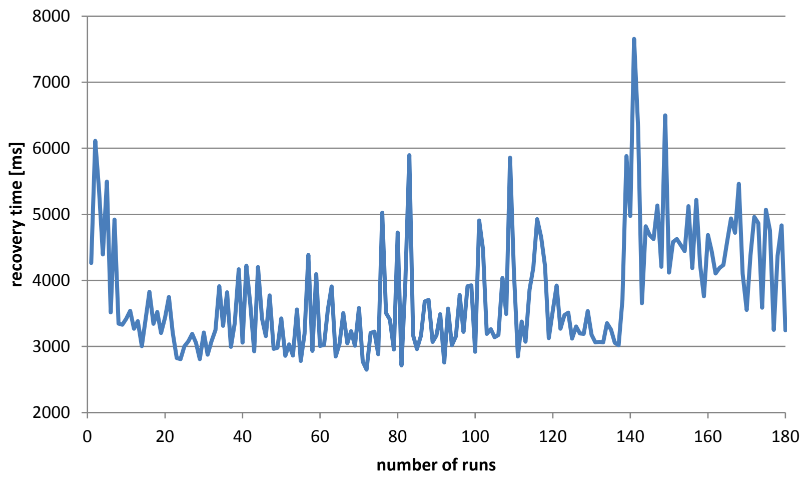
In this way we are transferring the media file chunk by chunk via the stream connection. These chunks are then stored in a cache and processed by the adaptation process. The adapted chunks are cached again and transferred to the next service. In case of a service failure, this service is replaced by 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 [11]). 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.

7.4.4. Service Recovery

The aim of the last evaluation was to measure the time needed to replace a failed service with a new one. We simulated the failure of the transcoding service and measured the time from the failure until the new service started sending out new data to the next neighbor in the workflow.

Figure 20 shows the results (in ms) for each recovery from a failure, simulating 180 service failures in total. The average time for recovery is about 3.8 s, including network delays and latency. Considering 1 MB (6 seconds) chunks our PUMA framework still can ensure a smooth playback (for more details see [11]).

Figure 20. Service recovery time.



7.5. Planetlab

In addition to our testbed, we ran some experiments on PlanetLab [43] to have a more common distribution of services. We were able to easily deploy some PUMA services on several selected servers within the PlanetLab network and run our workflow experiments after establishing the required environment. Most of these servers are usual single or dual core machines with around 2–3 GHz and 1–4 GB memory like the ones provided by the Massachusetts Institute of Technology and the University of California at Berkeley.

Due to the fact that the PlanetLab network is intensely used, we had to put up with a constant high degree of processing power utilization of around 60%–75% on every server. Despite the attempt to reserve and consequently increase some additional CPU time, the available processing power restrained the overall performance conspicuously. Nevertheless, we were able to successfully simulate some shorter workflows in this widely distributed and multiply used network.

As anticipated, the considerably lower computing power due to slower and multiply shared servers, compared to our environment, led to a significant increase of startup and delivery times, especially for longer workflows and even with increased CPU priority. These experiments did not allow multimedia streaming in real-time. Hence, these results substantiate our assertion that P2P networks are not suitable for service oriented multimedia stream processing.

8. Conclusion and 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. In contrast to common belief, we were able to show that content adaptation under soft real-time constraints is possible in a service-oriented way. Moreover, 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 200 ms as compared to 7 s 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 s movie using four services is 7 s. 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 3 s 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. To provide a continuous stream at the client side, timely replacement of failing services is needed. For this we implemented the PUMA E^2Mon monitoring protocol [10]. Our evaluations show that the Monitoring Service we implemented does not hamper the speed of the adaptation process, but allows replacing 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 3.8 s.

Although 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.

Acknowledgment

Part of this work was funded within the Emmy Noether Program of Excellence by the German Research Foundation (DFG).

References

1. Köhncke, B.; Balke, W.T. Preference-Driven Personalization for Flexible Digital Item Adaptation. *Multimedia Syst.* **2007**, *13*, 119–130.
2. Jannach, D.; Leopold, K. Knowledge-Based Multimedia Adaptation for Ubiquitous Multimedia Consumption. *J. Netw. Comput. Appl.* **2007**, *30*, 958–982.
3. PPLive. Available online: http://www.pplive.com (accessed on 18 March 2011).
4. Zattoo. Available online: http://www.zattoo.com (accessed on 18 March 2011).
5. Jannach, D.; Leopold, K.; Hellwagner, H. An Extensible Framework for Knowledge-Based Multimedia Adaptation. In *Proceedings of 14th Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems* (*IEA/AIE*), *Lecture Notes in Artificial Intelligence 2070*, Ottawa, Canada, 2004; pp. 434–444.

6. Heinzl, S.; Mathes, M.; Freisleben, B. A Web Service Communication Policy for Describing Non-Standard Application Requirements. In *Proceedings of the IEEE/IPSJ International Symposium on Applications and the Internet* (*Saint*), Turku, Finland, July 2008; pp. 40–47.
7. Heinzl, S.; Mathes, M.; Friese, T.; Smith, M.; Freisleben, B. Flex-SwA: Flexible Exchange of Binary Data Based on SOAP Messages with Attachments. In *Proceedings of IEEE International Conference on Web Services* (*ICWS'06*), Chicago, IL, USA, September 2006.
8. Balke, W.T.; Nahrstedt, K. Multimedia Service Composition: A Brave New Topic. In *Proceedings of ACM Multimedia Conference*, New York, NY, USA, October 2004; pp. 1–2.
9. Vetro, A. MPEG-21 Digital Item Adaptation: Enabling Universal Multimedia Access. *IEEE MultiMedia* **2004**, *11*, 84–87.
10. Balke, W.T.; Diederich, J. A Quality- and Cost-Based Selection Model for Multimedia Service Composition in Mobile Environments. In *Proceedings of IEEE International Conference on Web Services*, Chicago, IL, USA, September 2006; pp. 621–628.
11. Brunkhorst, I.; Tönnies, S.; Balke, W.T. Multimedia Content Provisioning Using Service Oriented Architectures. In *Proceedings of IEEE International Conference on Web Services*, Bejing, China, September 2008; pp.262–269.
12. Lopez, F.; Martinez, J.; Valdez, V. MM Content Adaptation within the Cain Framework via Constraints Satisfaction & Optimization. *Comput. Sci.* **2007**, *4938*, 149–163.
13. Zhang, X.; Liu, J.; Li, B.; Yum, T.S.P. Coolstreaming/Donet: A Data-Driven Overlay Network for Peer-to-Peer Live Media Streaming. In *Proceedings of INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, Miami, FL, USA, 2005.
14. Hei, X.; Liang, C.; Liang, J.; Liu, Y.; Ross, K.W. A Measurement Study of a Large-Scale P2P IPTV System. *IEEE Trans. Multimedia* **2007**, *9*, 1672–1687.
15. Li, B.; Xie, S.; Keung, G.Y.; Liu, J.; Stoica, I.; Zhang, H.; Zhang, X. An Empirical Study of the Coolstreaming+ System. *IEEE J. Sel. Areas Commun.* **2007**, *25*, 1627–1639.
16. Vu, L.; Gupta, I.; Liang, J.; Nahrstedt, K. Measurement of a Large-Scale Overlay for Multimedia Streaming In *Proceedings of the 16th International Symposium on High Performance Distributed Computing*, Monterey, CA, USA, 2007.
17. Decneut, S.; Hendrickx, F.; Nachtergaele, L.; Assche, S.V. Targeting Heterogeneous MM Environments with Web Services. In *Proceedings of IEEE International Conference on Web Services*, San Diego, CA, USA, June 2004.
18. SOAP Message Transmission Optimization Mechanism. Available online: http://www.w3c.org/-TR/soap12-mtom (accessed on 18 March 2011).
19. Lam. G.; Rossiter, D. Streaming Multimedia Delivery in Web Services Based E-Learning Platforms. In *Proceedings of Seventh IEEE International Conference on Advanced Learning Technologies*, Niigata, Japan, July 2007; pp. 706–710.
20. Heinzl, S.; Seiler, D.; Juhnke, E.; Stadelmann, T.; Ewerth, R.; Grauer, M.; Freisleben, B. A Scalable Service-Oriented Architecture for Multimedia Analysis, Synthesis, and Consumption. *Int. J. Web Grid Serv.* **2009**, *5*, 219–260

21. Jordan, D.; Evdemon, J.; Alves, A.; Arkin, A.; Askary, S.; Barreto, C.; Bloch, B.; Curbera, F.; Ford, M.; Y. Goland, Y.; *et al. Web Services Business Process Execution Language Version 2.0*. Available online: http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf (access on 22 March, 2011).
22. van der Aalst, W.M.P.; Aldred, L.; Dumas, M.; ter Hofstede, A.H.M. Design and Implementation of the YAWL System. *Adv. Inf. Syst. Eng.* **2004**, *3084*, 281–305.
23. Amielh, M.; Devillers, S. Bitstream Syntax Description Language: Application of XML-Schema to Multimedia Content Adaptation. *IEEE Trans. Multimedia* **2005**, *7*, 463–470.
24. Amoretti, M.; Conte, G.; Reggiani, M.; Zanichelli, F. Designing Grid Services for Multimedia Streaming in an E-Learning Environment. *Concurr. Comput.: Prac. Experience* **2006**, *18*, 911–923.
25. Cabral, L.; Domingue, J.; Galizia, S.; Gugliotta, A.; Tanasescu, V.; Pedrinaci, C.; Norton, B. IRS-III: A Broker for Semantic Web Services Based Applications. In *Proceedings of 5th International Semantic Web Conference*, Athens, GA, USA, 2006.
26. Martin, D.; Burstein, M.; Hobbs, J.; Lassila, O.; McDermott, D.; McIlraith, S.; Narayanan, S.; Paolucci, M.; Parsia, B.; Payne, T.; Sirin, E.; Srinivasan, N.; Sycara, K. OWL-S: Semantic Markup for Web Services. Available online: http://www.w3.org/Submission/OWL-S/(access on 22 March 2011).
27. Chinnici, R.; Moreau, J.J.; Ryman, A.; Weerawarana, S. Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. Available online: http://www.w3.org/TR/wsdl20/ (access on 22 March 2011).
28. Fensel, D.; Lausen, H.; Polleres, A.; de Bruijn, J.; Stollberg, M.; Roman, D.; Domingue, J. *Enabling Semantic Web Services—The Web Service Modeling Ontology*; Spriger: New York, NY, USA, 2006; p. 41.
29. Paolucci, M; Kawamura, T.; Payne, T.R.; Sycara, K.P. Semantic Matching of Web Services Capabilities. In *Proceedings of ISWC '02 Proceedings of the First International Semantic Web Conference on the Semantic Web,* Sardinia, Italy, 2002.
30. Foster, H.; Uchitel, S.; Magee, J.; Kramer, J. Model-Based Analysis of Obligations in Web Service Choreography. In *Proceedings of IEEE International Conference on Internet and Web Applications and Service,* Gosier, French Guadeloupe, Febuary 2006; p.149.
31. Dignum, F. *Advances in Agent Communication.* International Workshop on Agent Communication Languages: Melbourne, Australia, 2003.
32. Milner, R. *Communication and Concurrency.* Prentice-Hall, Inc.: Upper Saddle River, NJ, USA, 1989.
33. Baldoni, M.; Baroglio, C.; Martelli, A.; Patti, V.; A Priori Conformance Verification for Guaranteeing Interoperability in Open Environments. *Comput. Sci.* **2006**, *4294*, 339–351.
34. Baldoni, M.; Baroglio, C.; Martelli, A.; Patti, V.; Schifanella, C. Verifying the Conformance of Web Services to Global Interaction Protocols: A First Step. In *Proceedings of international workshop on web services and formal method*, Versailles, France, 2005; pp. 257–271.
35. Levesque, H.; Reiter, R.; Lesp´erance, Y.; Lin, F.; Scherl, R. GOLOG: A Logic Programming Language for Dynamic Domains. *J. Log. Program.* **1997**, *31*, 59–83.

36. Berbner, R.; Spahn, M.; Repp, N.; Heckmann, O.; Steinmetz, R. Heuristics for QoS-Aware Web Service Composition. In *Proceedings of IEEE International Conference on Web Services*, Chicago, USA, September 2006; pp.72–82.
37. Nahrstedt, K.; Balke, W.T. Towards Building Large Scale Multimedia Systems and Applications: Challenges and Status. In *Proceedings of the first ACM international workshop on Multimedia service composition*, Hilton, Singapore, November 2005.
38. Liu, Y.; Ngu, A.; Zeng, L. QoS Computation and Policing in Dynamic Web Service Selection. In In *Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, Pittsburgh, PA, USA, 2004.
39. Jainand, M. Dovrolis, C. End-to-End Available Bandwidth: Meas-urement Methodology, Dynamics, and Relation with TCP Throughput. In *Proceedings of the 13th international World Wide Web Conference on Alternate Track Papers & Posters,* Manhattan, NY, USA, 2002.
40. PUMA. Available online: http://www.l3s.de/puma (accessed on 18 March 2011).
41. Broekstra, J.; Kampman, A.; van Harmelen, F. Sesame: A generic architecture for storing and querying rdf and rdf schema. In *Proceedings of the First International Semantic Web Conference on the Semantic Web,* Sardinia, Italy, June 2002.
42. Holzmann, G.J. *The SPIN Model Checker, Primer and Reference Manual*; Addison-Wesley: Boston, MA, USA, 2004.
43. PlanetLab. Available online: http://www.planet-lab.org (accessed on 18 March 2011).