Multimedia Content Provisioning using Service Oriented Architectures

Ingo Brunkhorst, Sascha Tönnies, Wolf-Tilo Balke
Research Center L3S
30167 Hannover, Germany
{brunkhorst, toennies, balke}@L3S.de

Abstract

Today, multimedia system are still widely realized as monolithic systems. But building such applications using Service-Oriented Architectures — especially for the Processing and Delivery of continous Multimedia data streams — has been a controversial topic since years. As a result, applications in the multimedia domain cannot yet benefit from Web service architectures. Thus, building and maintaining large-scale multimedia applications remains a difficult, costly, time-consuming and challenging problem.

In this paper we present our approach for building large-scale multimedia systems and compare it with the current state of the art, concentrating on the selection and validation of multimedia service composition.

1. Introduction

Although Web services architectures provide flexible solutions for many large-scale applications, the domain of multimedia systems is still widely dominated by monolithic implementations. But without the use of service-oriented approaches, building large-scale multimedia systems, e.g., for video-conferencing, IPTV or video on demand applications, remains a difficult, costly, and time-consuming problem.

The necessity to provide actual multimedia services first arose in the context of the i-mode framework [22], successfully launched in 1999 by the Japanese provider NTT DoCoMo. The technical innovation was tightly coupled with the deployment of a novel business model. The benefits of providing value-added services as a network provider are threefold. First, the quality of both services and content can be controlled and guaranteed by a trusted party. Second, QoS can be adequately measured and optimised beyond what an individual service provider can achieve. Third, the close connection between users and network providers, as well as network and content providers allow for simple and cost-effective billing solutions. Following this business model, currently many phone and television companies are pursuing "triple play" campaigns, trying to offer telephony, internet and television within one network infrastructure.

Looking at business processes at large, Web service architectures are now commonly used to build service chains instantiating individual workflows. Recent research mainly addressed the problem of dynamically selecting suitable services for composing workflows in a standardised manner. The actual composition then exchanged static data. This is entirely different in multimedia workflows. Here the basic workflows, e.g. to perform digital item adaptation (DIA), are generally well-understood. The difficulty in the multimedia domain is the nature of the data that has to be exchanged: it usually consists of continuous data streams dependent in time and space. And in fact, one of the main reasons for the lack of uptake from the multimedia community is still the limited and inadequate support for handling continuous data with Web services, especially media streaming support. Solving these problems enables the re-use of components for instance in different personalised adaptation processes.

In this paper we present our multimedia service architecture for the sample application of media streaming and specifically focus on the workflow selection with continuous data. Especially the validation of interoperability between services is an important part of the selection process. When considering which services to include in a service composition, or as in our case, into a content adaptation and delivery workflow, we have to make sure that the services not only provide a certain set of operations that can be invoked. More important is that services can interoperate with each other w.r.t complex protocols. We have to deal with the dynamic and timely replacement of failing services and the reconfiguration of the workflow to reflect changes in the network or client device. For this, a semantic rich description of the services and the involved protocols is needed.

In the course of this paper we will present our approach for multimedia service composition validation. Extending techniques from the multi-agent systems domain [3, 5] we show how to define conversation policies and protocols. We verified our approach with a prototypical implementation of a media streaming testbed, distributed over several servers. We evaluated our approach regarding the selection of services, the efficiency of interoperability validation, and the recovery time from a service failure. We show that our approach is resilient and efficient enough to allow real time digital item adaptation.

In the next section we will start with a quick overview of our architectural design, followed by related work (section 3). Section 4 will give a description of our approach of planning, creating and validation service compositions for multimedia workflows. This is followed by a short evaluation of our test prototype system (section 5) and conclusions (section 6).

2 Architecture Overview

Figure 1 gives an overview of our service oriented architecture. The three central components of the architecture are the *decision engine*, *validation engine* and the *execution engine*.

2.1 Decision Engine

The first step necessary is the creation of a suitable workflow for the task at hand, for instance a media content adaptation process.

Workflows. Workflows are sequences of individual processing 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 2). For instance, an adaptation workflow may include roles for transcoding, scaling, bit-depth reduction and audio re-encoding.

Since for multimedia applications the involved workflows are generally well understood anyway, complex reasoning frameworks for workflow construction are not necessary. All necessary workflows can be engineered by multimedia specialists and specifically tailored towards the needs of the required application. The selection of which workflow is most suitable, is based on rules and service parameters such as projected service costs and QoS constraints.

However, compositions on continuous data (possibly with QoS-based models) are not supported by the currently available orchestration engines for Web service composition. Actually, there are first proposals to extend BPEL to support such functionality [7], but they still lack implementation.

2.2 Validation Engine

After the creation of some workflow, a set of suitable services needs to be found for the different roles. As common in Web service architectures, discovery of these services is done using a UDDI-like central service registry and for instance OWL-S descriptions.

Validation of Service Interoperability. Before a workflow can be started, it must be checked whether the involved participants are able to fill the assigned role and interoperate with each other. As main focus of this paper, a detailed description of the techniques for interoperability checking is given in section 4.

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 other parameters depending on device capabilities or the execution environment [6, 16]. All the costs can than be aggregated to 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.

2.3 Execution Engine

Once the service orchestration is prepared, i.e. all roles in the workflow have been filled with a suitable service, the service chain has to be executed.

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 order and are able to dynamically substitute failing services with other service implementations. Since Web services are stateless, to handle continuous data introducing a session concept for each stream into our architecture was necessary.

Monitoring of Services. The special role of the monitoring service stems from the dynamic properties of streaming over unreliable channels, like the Internet or mobile networks. Monitoring services includes the checking the current state of a service and also the timely replacement of failing services. Other types of workflow management includes the replacement of services when the network parameters, e.g. bandwidth or latency, change such that a certain service or workflow is no longer suitable.

3 Related Work

In the Web service business world, business process execution languages (like e.g. BPEL [15], YAWL [23], or 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.

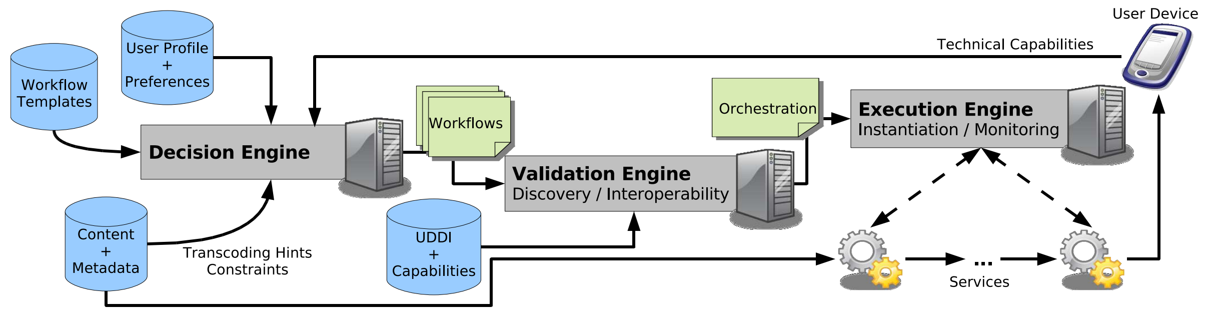

Figure 1. Architecture Overview

The main difference to general purpose languages is the inclusion of continuous data streams in our workflows. There exist only a few extensions to Web service description languages to support continuous data streams. Again, none of them can be considered ready to use yet [17, 1, 2].

Moreover, the workflows currently needed by our architecture are simple compared to the flexibility of the modelling constructs available in the general purpose workflow frameworks. As a result, we use our own RDF-based workflow model and language with added support for describing the streaming of data between participants. For the actual instantiation there already exist systems, like IRS-III [9], 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.

Selection For describing the capabilities of our multimedia services, OWL-S [19] 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 [10]. 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 in [12, 21].

Validation To our knowledge, the only system to validate interoperability between Web services, was developed by Foster et al. [13]. 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.

But the checking of interoperability between services is also similar to problems faced in multi-agent systems domain. Generally, multiple ways exist to check if interoperability between agents is possible. Recent work especially focusses on checking the conformance of policies w.r.t. global communication protocols [11]. For the actual conformance tests, different approaches can be used. Popular instances are bisimulation [20] (or a variant of it [3]), as well as the checking of execution traces [4].

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 modelling of actions in form of IOPE objects, specifying *Inputs*, *Outputs*, *Preconditions* and *Effects*. Instances of such languages are OWL-S, WS-CDL, BPEL, WSMO [12], or systems like IRS-III [9]. Respective languages from the multi-agent systems domain are given in [11] with a foundation in Action Logic, for instance systems like Golog [18].

4 A new approach to selection and validation of workflows

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

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 as an example media streaming. It consists of fetching, adapting and delivering the content data to the user. Each step, or *role* (see figure 3) 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 bandwith codec used by mobile phones, like h.264.

In our architecture, the *multimedia workflows* are represented using a straightforward action-like model, consisting

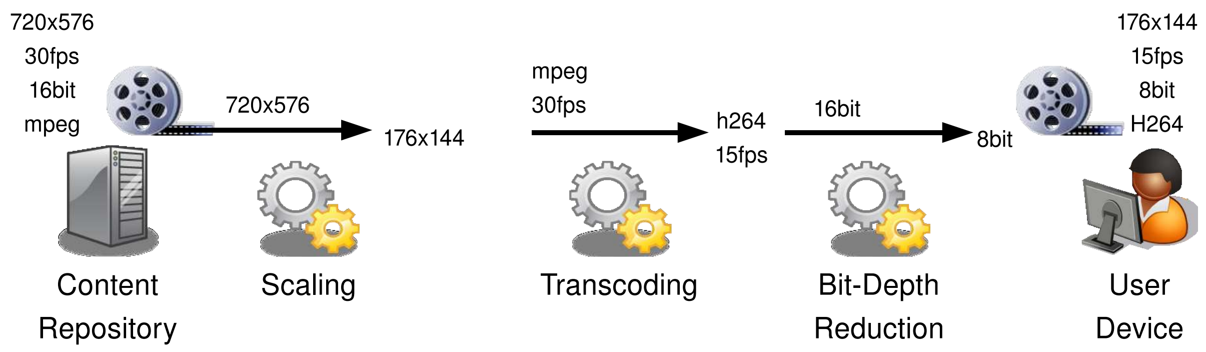

Figure 2. Example adaptation workflow for a mobile device

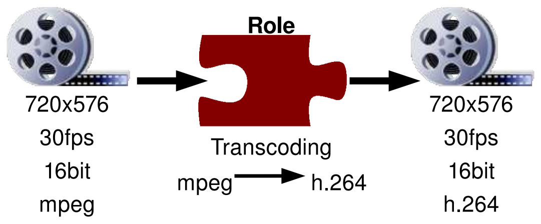

Figure 3. Role Example: Transcoding a stream from one format to another

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 the 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 optimizied for quality, network efficiency or costs. As a result the system ends up with a ranked list of suitable workflows. Figure 2 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.

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" [19]). Figure 4 shows a part of the OWL-S Description of an available scaling service.

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

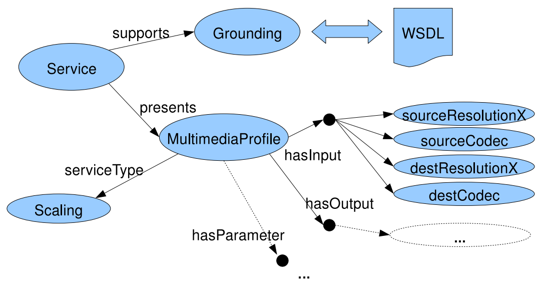

Figure 4. OWL-S Description of a scaling service (profile sub-graph)

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.

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 behaviour (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 behaviour of services prior or later in the chain is essential.

For describing these *capabilities* of a service that lies beyond the observable behaviour (as provided in the WSDL and OWL-S descriptions) additions to the Web service description languages are needed, e.g. WS-CDL+C [5]. In multi agent systems [11], there exist 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 behaviour of a service.

For multimedia workflows, a service is interoperable with other services in the workflow, if the messages it is expecting to receive or sending is in line with the expected behaviour 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 re-establishing 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 behaviour for certain roles in the workflow.

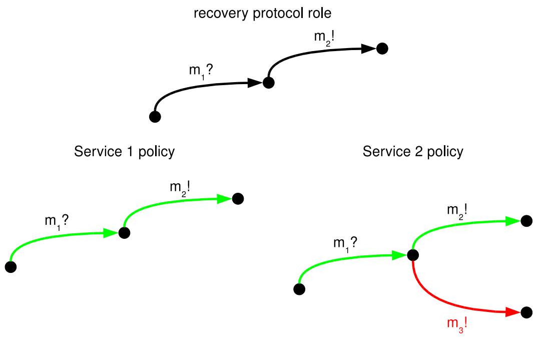

Figure 5. Recovery: protocol specification and service policies

Figure 5 gives an example for one of the service interactions for recovering from a failed service, m? represent incoming messages, m! outgoing messages.

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.

For a different part of the protocol, to temporary pause a stream (see figure 6), 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.

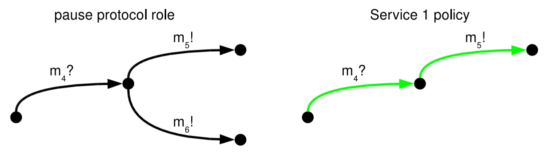

Figure 6. Stop: protocol specification and service policies

Protocol specifications also include the expected interactions following an outgoing message, as shown in figure 7.

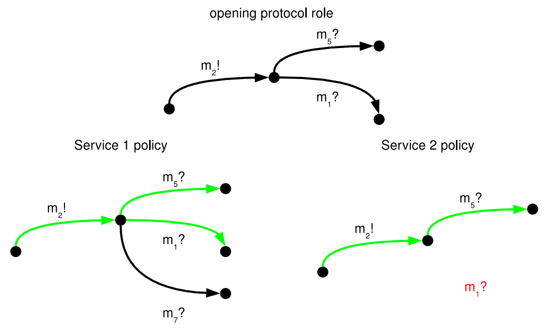

Figure 7. Opening: Protocol Specification and Service Policies

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. Different then service 1, which in addition also listens for a diagnostics message m_7?, which is not required by the protocol, and so does not invalidate interoperability.

5 Implementation and Evaluation

We implemented the framework as described in section 2. The implementation was done in Java, including native C code calls to libraries necessary for stream processing. The Apache CXF Web service framework was used for the implementation of the Web service infrastructure. If not reported otherwise, all measurements are averages over a hundred independent runs and were performed an IBM T43 with 1.5GB of memory and a 2.0 GHz Intel Pentium M processor.

Selection of Services Given the workflow, the first step in creating the service composition is the selection of suitable services from the service registry. We require OWL-S in addition to WSDL for describing our services. Our service registry is realized using the open-source RDF store Sesame [8], 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 service type (e.g. transcoding) and the available operations of each service.

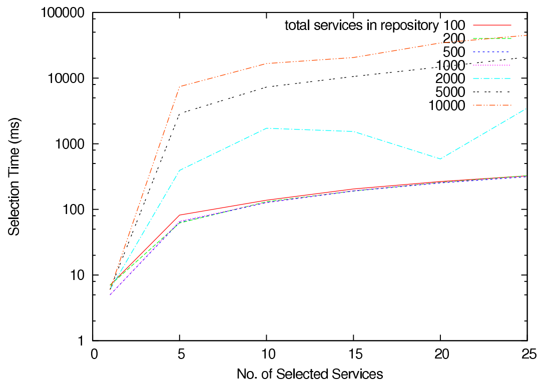

Figure 8. Selecting service candidates from the repository

Figure 8 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 sec 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 < 200ms). However, time for retrieving the service description from the sesame database is considerably increasing 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.

Validation For modelling Web service choreographies and compositions, there exist a number of description languages, like WSMO and 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 4 in many of these languages. At the current state, we specify protocol roles already at the workflow level, using elements from WS-CDL [5] for simplicity.

Describing the policy, i.e. the behaviour 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 it's current state, checking interoperability is done according to the description in [3], which is a variant of bisimulation [20].

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.

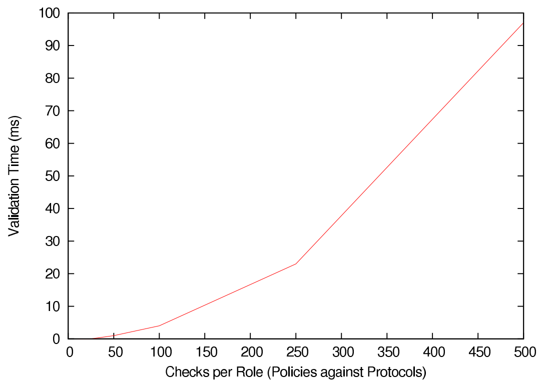

Figure 9. Validating service policies against the protocol role

Figure 9 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 ranges in the area of 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-

700ms). Also, this materialization can already be performed for several selected services at an earlier step, e.g. during service selection.

Adaptation Services For implementing the adaptation services we used ffmpeg (`http://ffmpeg.mplayerhq.hu/`) to facilitate the content processing. The transport of content data to the ffmpeg process is done over unix pipes. We also implemented services based on mencoder, jmf and jvlc, deciding for ffmpeg because it supports a large number of codecs and processing options. An adaptation Web service (figure 10) is internally implemented as a pipeline working on queues containing the multimedia data, stored in chunks of variable size. The reader stores the received data into the input queue, the worker takes the newly received chunks for processing with the external process and stores the result into the output queue. Those chunks are then send to the next service in the workflow by the writer. The purpose of the Web service interface is the handling of the stream control protocol, notifications and monitoring.

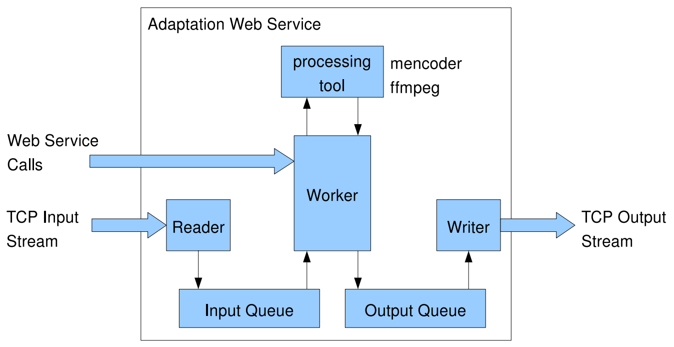

Figure 10. Implementation of an adaptation service

Recovery from service failure The aim of the last evaluation was to measure the time needed to replace a failed service with a new one. In an instantiated workflow based on the above example, we simulated the failure of one intermediate service (in our case the transcoding service) and measured the time from the failure until the replacement service started sending out new data to the next neighbour in the workflow chain, i.e. the bit-depth reduction service.

Figure 2 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 seconds, including network delays and latency. In face of buffering strategies prevalent in todays multimedia streaming frameworks this is enough for seamless recovery.

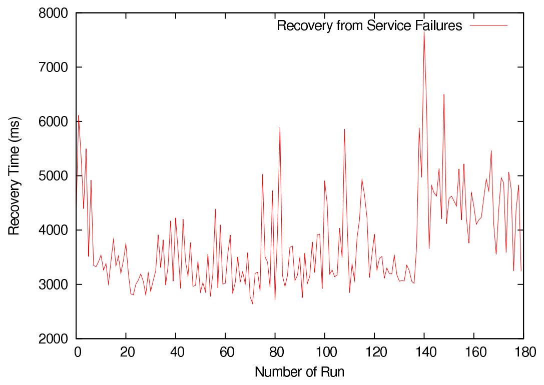

Figure 11. Measured times for replacing the transcoding service

6 Conclusion

In this paper we introduced our design of a semantic Web service framework for multimedia content adaptation. We specifically presented the workflow creation, service selection and validation parts of our architecture. Currently, support for combining Web services and stream-based multimedia applications is severly limited, due to multiple factors. First, the lack of open implementations for multimedia stream processing, in Java or other languages. Second, support for integrating stream-based content into the Web service infrastructure is also missing.

In this paper we largely focus on interoperability issues when dealing with complex stream control and monitoring protocols in a heterogenous environment. Checking whether the behaviour of a service follows the required protocol is essential, especially when services can fail and need to be replaced in a timely manner. It must be ensured that they are able to receive and process, as well as send all messages necessary to complete their role. Based on the implementation of our architecture we performed experiments in practical scenarios to asses the performance of our approach. We were able to show that the overall times for locating and selecting up to 5 candidates for a role in the workflow is practical with about 3 sec. The major part of this time is used for retrieving the service description from a sesame database; this process has to be severly speed up by using adequate indexes. The actual time needed for validating the policies of a service against the protocol specification of a workflow role ranges in the area of only milliseconds. For the on-the-fly replacement of failed services, we could show, that the average time needed for recovery is only about 3.8 seconds, including network delays and latency. For the applications in todays multimedia streaming frameworks these results are promising to allow for a service-oriented processing of continuous stream data.

References

[1] M. Amielh and S. Devillers. Bitstream Syntax Description Language: Application of XML-Schema to Multimedia Content Adaptation.

[2] M. Amoretti, G. Conte, M. Reggiani, and F. Zanichelli. Designing grid services for multimedia streaming in an e-learning environment. *Enabling Technologies: Infrastructure for Collaborative Enterprises, 2004. WET ICE 2004. 13th IEEE International Workshops on*, pages 331–336, 14-16 June 2004.

[3] M. Baldoni, C. Baroglio, A. Martelli, and V. Patti. A priori conformance verification for guaranteeing interoperability in open environments. In *Proceedings of the 4th International Conference on Service Oriented Computing, ICSOC 2006*, volume 4294 of *Lecture Notes in Computer Science*, pages 339–351, Chicago, USA, December 2006. Springer.

[4] M. Baldoni, C. Baroglio, A. Martelli, V. Patti, and C. Schifanella. Verifying the conformance of web services to global interaction protocols: A first step. In M. Bravetti, L. Kloul, and G. Zavattaro, editors, *EPEW/WS-FM*, volume 3670 of *Lecture Notes in Computer Science*, pages 257–271. Springer, 2005.

[5] M. Baldoni, C. Baroglio, A. Martelli, V. Patti, and C. Schifanella. Interaction protocols and capabilities: A preliminary report. In J. J. Alferes, J. Bailey, W. May, and U. Schwertel, editors, *PPSWR*, volume 4187 of *Lecture Notes in Computer Science*, pages 63–77. Springer, 2006.

[6] W.-T. Balke and J. Diederich. A quality- and cost-based selection model for multimedia service composition in mobile environments. In *ICWS*, pages 621–628. IEEE Computer Society, 2006.

[7] B. Biornstad, C. Pautasso, and G. Alonso. Control the Flow: How to Safely Compose Streaming Services into Business Processes. pages 206–213. IEEE Computer Society Washington, DC, USA, 2006.

[8] J. Broekstra, A. Kampman, and F. van Harmelen. Sesame: A generic architecture for storing and querying rdf and rdf schema. In Horrocks and Hendler [14], pages 54–68.

[9] L. Cabral, J. Domingue, S. Galizia, A. Gugliotta, V. Tanasescu, C. Pedrinaci, and B. Norton. Irs-iii: A broker for semantic web services based applications. In I. F. Cruz, S. Decker, D. Allemang, C. Preist, D. Schwabe, P. Mika, M. Uschold, and L. Aroyo, editors, *International Semantic Web Conference*, volume 4273 of *Lecture Notes in Computer Science*, pages 201–214. Springer, 2006.

[10] R. Chinnici, J.-J. Moreau, A. Ryman, and S. Weerawarana. Web services description language (wsdl) version 2.0 part 1: Core language. Technical report, World Wide Web Consortium (W3C), June 2007.

[11] F. Dignum, editor. *Advances in Agent Communication, International Workshop on Agent Communication Languages, ACL 2003, Melbourne, Australia, July 14, 2003*, volume 2922 of *Lecture Notes in Computer Science*. Springer, 2004.

[12] D. Fensel, H. Lausen, A. Polleres, J. de Bruijn, M. Stollberg, D. Roman, and J. Domingue. *Enabling Semantic Web Services – The Web Service Modeling Ontology*. Springer, 2006. Not yet published. Available: October 15, 2006.

[13] H. Foster, S. Uchitel, J. Magee, and J. Kramer. Model-based analysis of obligations in web service choreography. In *AICT/ICIW*, page 149. IEEE Computer Society, 2006.

[14] I. Horrocks and J. A. Hendler, editors. *The Semantic Web - ISWC 2002, First International Semantic Web Conference, Sardinia, Italy, June 9-12, 2002, Proceedings*, volume 2342 of *Lecture Notes in Computer Science*. Springer, 2002.

[15] D. Jordan, J. Evdemon, A. Alves, A. Arkin, S. Askary, C. Barreto, B. Bloch, F. Curbera, M. Ford, Y. Goland, A. Guzar, N. Kartha, C. K. Liu, R. Khalaf, D. Knig, M. Marin, V. Mehta, S. Thatte, D. van der Rijn, P. Yendluri, and A. Yiu. Web services business process execution language version 2.0. Technical report, OASIS, November 2007.

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

[17] G. Lam and D. Rossiter. Streaming multimedia delivery in web services based e-learning platforms. In J. M. Spector, D. G. Sampson, T. Okamoto, Kinshuk, S. A. Cerri, M. Ueno, and A. Kashihara, editors, *ICALT*, pages 706–710. IEEE Computer Society, 2007.

[18] H. Levesque, R. Reiter, Y. Lespérance, F. Lin, and R. Scherl. GOLOG: A logic programming language for dynamic domains. *Journal of Logic Programming*, 31:59–84, 1997.

[19] D. Martin, M. Burstein, J. Hobbs, , O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, E. Sirin, N. Srinivasan, and K. Sycara. Owl-s: Semantic markup for web services. Technical report, World Wide Web Consortium (W3C), 2004.

[20] R. Milner. Communication and concurrency. Technical report, Upper Saddle River, NJ, USA, 1989.

[21] M. Paolucci, T. Kawamura, T. R. Payne, and K. P. Sycara. Semantic matching of web services capabilities. In Horrocks and Hendler [14], pages 333–347.

[22] K. Tsujimura. Docomo's challenge towards new mobile services. page 876.

[23] W. M. P. van der Aalst, L. Aldred, M. Dumas, and A. H. M. ter Hofstede. Design and implementation of the yawl system. In A. Persson and J. Stirna, editors, *CAiSE*, volume 3084 of *Lecture Notes in Computer Science*, pages 142–159. Springer, 2004.