A Real-time Service Oriented Infrastructure

Dimosthenis Kyriazis, Andreas Menychtas, George Kousiouris
National Technical University of Athens
Athens, Greece
{dimos, ameny, gkousiou}@mail.ntua.gr

Karsten Oberle, Thomas Voith
Alcatel Lucent
Stuttgart, Germany
{karsten.oberle, thomas.voith}
@alcatel-lucent.com

Michael Boniface
University of Southampton IT Innovation Centre
Southampton, UK
mjb@it-innovation.soton.ac.uk

Eduardo Oliveros
Telefonica Investigation e Disarollo
Madrid, Spain
eod@tid.es

Tommaso Cucinotta
Scuola Superiore Sant'Anna
Pisa, Italy
t.cucinotta@sssup.it

Sören Berger
University of Stuttgart
Stuttgart, Germany
soeren.berger@rus.uni-stuttgart.de

Abstract — **Service oriented environments and real-time systems have been two mutually exclusive technological areas. Taking into consideration the main concepts of service orientation, significant challenges exist in providing and managing the offered on-demand resources with the required level of Quality of Service (QoS), especially for real-time interactive and streaming applications. In this paper we propose an approach for providing real-time QoS guarantees by enhancing service oriented infrastructures with coherent and consistent real-time attributes at various levels (application, network, storage, processing). The approach considers the full lifecycle of service-based systems including service engineering, Service Level Agreement (SLA) negotiation and management, service provisioning and monitoring. QoS parameters at application, platform and infrastructure levels are given specific attention as the basis for provisioning policies in the context of temporal constraints. We also demonstrate through use cases the need for real-time scheduling as a fundamental process to provide QoS guarantees.**

Keywords – real-time; service oriented infrastructure; cloud computing; quality of service;

I. INTRODUCTION

Service Oriented Architectures (SOAs) [1] refer to a specific architectural paradigm that emphasizes implementation of components as modular services that can be discovered and used by clients. Infrastructures based on the SOA principles are called Service Oriented Infrastructures (SOIs). Through the agility, scalability, elasticity, rapid self-service provisioning and virtualization of hardware, Service Oriented Architecture principles are reflected into Clouds, which provide the ability to efficiently adapt resource provisioning to the dynamic demands of Internet users. Many architectural paradigms from distributed computing such as service-oriented infrastructures, Grids and virtualization are incorporated into Clouds. There are three main classes in the cloud services stack which are generally agreed upon [2]:

- Infrastructure as a Service (IaaS), which refers to the provision of 'raw' machines (servers, storage, networking and other devices) on which the service consumers deploy their own software (usually as virtual machine images).
- Platform as a Service (PaaS), which refers to the provision of a development platform and environment providing services and storage, hosted in the cloud.
- Software as a Service (SaaS), which refers to the provision of an application as a service over the Internet or distributed environment.

In this paper, we do not focus on a specific class of the aforementioned ones, but describe how real-time aspects are addressed across the classes. The proposed approach has been developed in the EU-funded project IRMOS [3], targeting soft real-time applications that have stringent timing and performance requirements, but for which some violations of the timing constraints are acceptable provided these are well understood and carefully managed, as they lead to degradation in the provided QoS level. Besides the approach, a set of tools and methodologies [5], [6] have been implemented offering the corresponding functionality in the IaaS, PaaS and SaaS classes.

The remainder of the paper is structured as follows: Section II gives an overview of the proposed service oriented infrastructure including the key features and a high-level view of the architecture, while Section III introduces the control loops concept that allows the infrastructure to provide QoS guarantees. The "implementation" of the control loops refers to specific processes (also called channels) that enable and guarantee real-time, presented in Sections IV and V correspondingly. Section VI includes specific use cases that

The research leading to these results has been supported by the European Commission under grant agreement n.214777, in the context of the IRMOS Project. More information at: http://www.irmosproject.eu.

demonstrate the importance of scheduling periods and of the real-time scheduler in one of the aforementioned processes / channels in the proposed approach to provide QoS guarantees for real-time interactive multimedia applications. The paper concludes with a discussion on future research and potentials for the current study.

II. SERVICE ORIENTED INFRASCTRUCTURE

A. *Key Features*

The proposed architecture adopts a service-oriented approach to allow services to interact dynamically and continuously, spanning between different domains, and ranging from the application level down to the level of network resources management and the execution environment.

To achieve the real-time functionality and the required QoS level, the infrastructure operation is separated in two phases: the offline, where the application and Application Service Components (ASCs) are prepared (i.e. development, modeling, etc) and the online, where the resources are negotiated and reserved and the application is initialized and operates. Expanding on this in greater detail as depicted in the following figure (Figure 1):

- Offline Phase (design-time service engineering): This phase includes the processes for developing / adapting application components to the SOI and the creation of descriptors and documents for the application operation such as models, mapping rules, initialization scripts, SLA templates and workflow descriptions.
- Online Phase (negotiation, execution and monitoring): This phase begins with the SLA negotiation and as soon as the SLAs are agreed (signed), the IaaS provider reserves the resources (computational, storage and network) for use within the requested time interval. When the execution of the application starts, the PaaS provider is responsible for orchestrating and monitoring, until completion, the workflow execution. At any time during the execution, an exception and / or SLA violation might occur; as a reaction mechanisms to adapt the resources (e.g. live migration) are put in place while re-negotiation of SLAs may be triggered in order to re-guarantee the QoS provision of the application and the application service components.

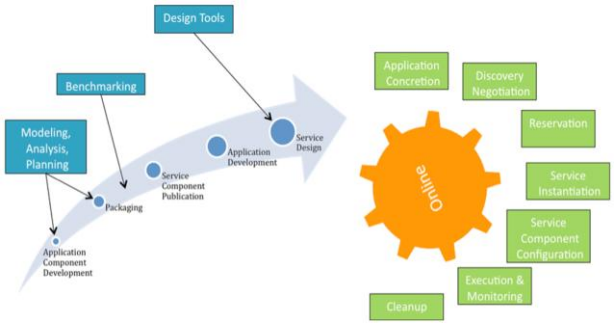

Figure 1. Two-Phases Approach

B. *Architecure*

Based on the cloud service models, in this section we briefly discuss the overall architecture (details can be found in [7]) in order to describe how real-time is achieved across these service models. In the SaaS service model, a specific methodology and tools have been developed, which allow application developers to engineer their application to deploy it within the SOI [6]. The PaaS service model operates between applications and virtualized resources. As shown in the following figure (Figure 2), the core elements are Service Engineering and Service Management, which are described in more detailed in the subsequent sections. This layer aims to provide and manage the execution of real-time services inside the IaaS on request of the Application Layer, while conforming to the real-time constraints as determined in the Application-SLA. Apart from managing applications execution, the framework supports service engineering, fully automated SLA negotiation and re-negotiation, mapping high level performance parameters to low level resource parameters, discovery and reservation of the ISONI resources needed for the execution. During the execution phase of the application, the PaaS provider monitors continuously and manages the application components and the resources either directly through the application wrappers or through the monitoring interface of the IaaS layer.

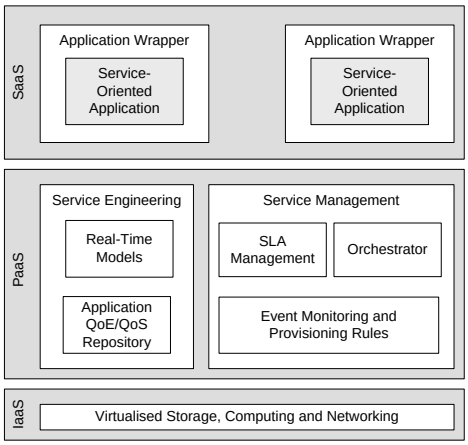

Figure 2. High-level Service Oriented Architecture

III. CONTROL LOOPS

In this section we introduce the concept of control loops which are implemented with processes being employed to support application provisioning and execution through the virtualized execution environment and networking infrastructure. To achieve this, the platform does not merely provide a set of services but also cross layer workflows that consider the control channels and information exchanges which are required to support real-time management of interactive applications throughout the full lifecycle.

In order to minimize manual configuration and deliver on-demand QoS-aware services, all subsystems are self-managed and reconfigured in order to achieve management efficiencies, and to react on QoS failures (such as an SLA violation or network link failure) in a timely way. To achieve the latter, we

introduce three control loops that are all at technical level and provide the necessary functionality in order to maintain QoS metrics across the architectural levels. The Control Loops are the following and are depicted in Figure 3:

- Application Control: It deals with the relationship between users and applications required to guarantee the application QoS. This control loop is managed by the application itself and the application developer in response to either user events or platform events. It is implemented with the use of models, workflows and tools that produce artifacts capturing the applications' behavior and estimating resource needs in advance of execution. During runtime it refers to application monitoring that may for example trigger events or require for changes in the provided resources.
- Environment Control: It deals with the relationship between applications and virtual resources in order to guarantee the platform QoS, as agreed in the SLAs. This control loop is managed by the platform services in response to application and virtualisation events. It is implemented by the framework services (set of tools) that support and manage the applications at run-time (e.g. actions triggered if either the application or resources do not perform as expected or need to be adjusted).
- Virtualization Control: It deals with the relationship between virtual and physical resources in order to guarantee the infrastructure QoS. This control loop is managed within the IaaS provider in response to platform or physical events. It is implemented by intelligent networking services and tools as well as by the Execution Environment for computing and storage services.

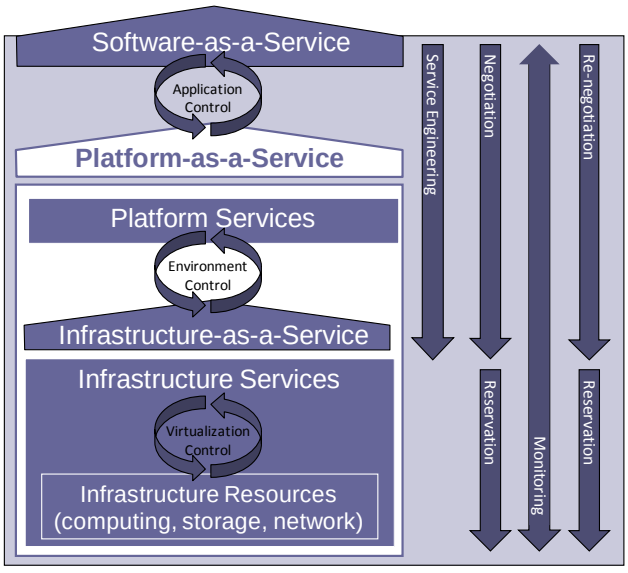

Figure 3. Control Loops

The actual implementation of the control loops refers to tools and services used on different levels in order to monitor the applications' execution, communicate possible events and take corrective actions if needed. We focus on five main processes / channels implementing the control loops: Service Engineering, Negotiation, Reservation, Monitoring and Re-negotiation. These processes as well as their orientation towards the control loops are analyzed in the following paragraphs of this paper (while the first three are considered to be enablers for real-time, the latter two are "facilitators" of real-time and interactivity as explained later in this paper).

IV. Processes Enabling Real-Time

The goal of the channels / processes described in this section is to provide the necessary information prior to execution in order to describe and model the applications, predict their behavior, negotiate and reserve resources according to the aforementioned modeling and prediction in order to ensure that during execution the requested level of quality will be offered by the infrastructure.

A. *Service Engineering*

The goal of the Service Engineering process is to estimate the resources required for an application and identify the QoS parameters that have critical influence on the application's performance. The main actor in this process is the Application Developer, who uses a set of tools to provide the necessary information to the platform services in order to estimate the required resources for the application execution. These tools are the following:

1) Service Modelling Environment: A dedicated tool that contains a profile for modelling ASCs using UML2. The Application Developer uses this tool to model the application and specify a number of parameters that are necessary for the effective deployment of an ASC, and an application in general. These include workload parameters that affect the performance requirements of the ASC (e.g. number of users connected to an eLearning application) and metrics that are used in order to quantify the level of QoS offered by the platform for this ASC (e.g. response time of the eLearning real-time server **Error! Reference source not found.**). The outcomes of this process are the Application Service Component Description (ASCD). The modelling environment automatically produces the Application-SLA Template by combining the individual ASCDs of the components. The Application-SLA is then published by the SaaS Provider to the PaaS Provider.

2) Mapping Service: A service providing an Artificial Neural Network-based rule / model, that depicts the relationships between the ASC characteristics / inputs, the different hardware configurations and the resulting QoS levels. It connects high level application workload features (such as number of users, resolution of processed images etc.) with application QoS requirements (like the achieved frames per second, application response time etc) and low level resource parameters. Through these rules, the platform provider can observe the effect of selected resources on the QoS output for a given execution with specific workload

parameters. An initial implementation of this appears in **Error! Reference source not found.**.

3) Performance Estimation Service: A service using the mapping rules in conjunction with modelling approaches such as Finite State Machines (FSMs) and Discrete Event Simulation (DES) in order to include workflow, events, interactivity, uncertainty and optimization in the ASC and application performance models, along with probabilistic guarantees.

The functionality of the Control Loops applied to the Service Engineering process refers to:

- Application Control: The Application Performance Models allow for modeling user demand in respect to application QoS. Optimization and updates of the models is feasible through a feedback loop that provides updated runtime information (received from the Monitoring service).
- Environment Control: The set of low level resource requirements that is being produced by the Performance Estimation Service allows for modeling application QoS in respect to virtual resources. The updates that may occur on the application control loop can be "passed" to the environment control through the updated model or updated ASCD parameters that will be reflected in an updated set of the resource requirements.

B. Negotiation

The goal of the Negotiation process is to agree the Application SLAs (considering customer requirements) and Technical SLAs (discovery of resources available, which fulfill the customer requirements) between the customer and the corresponding providers in order to proceed with the reservation of the resources according to the agreements [16]. The prerequisites for the Negotiation process are the Application-SLA Template and the Rules (from the Mapping Service) as produced by the Service Engineering process.

There are many actors in this process, namely the Customer, the SaaS Provider, the PaaS Provider and the IaaS Provider, since there are two different negotiations involved: (i) Application-SLA negotiation between the Customer and the SaaS Provider, and (ii) Technical-SLA negotiation between PaaS Provider and IaaS Provider. The tools / services engaged in this process are the following:

1) SLA Negotiator: A service orchestrating the negotiation process and providing valid SLA offers to the Customer prior to the execution of the services. It represents the central component during the SLA negotiation process.

2) A-SLA Manager: A service responsible for the management of Application-SLAs, which includes query, publishing, creation and update SLA templates and mapping commitments to IaaS resources.

3) Performance Estimation Service: During the negotiation phase, this service provides resource specification descriptions, which encompass information related to the Virtual Machine Units (VMUs) and the network links interconnecting them. This information includes QoS annotations as requests towards the IaaS providers.

4) Discovery Service: A service responsible for registering available IaaS Providers that meet the QoS parameters defined in the Application-SLA. It is designed so as to store the pricelists of the IaaS providers and to retrieve them when contacted by the SLA Negotiator. The service also includes a function that is used by the IaaS providers to advertise their capabilities with QoS properties.

5) T-SLA Manager: A service responsible for the management of Technical-SLAs which specify resources procured with IaaS providers. Technical-SLAs are offered by the IaaS providers as a response to requests from the PaaS providers. One of the key functionalities of the T-SLA Manager is reporting any SLA violations to the SaaS provider through a notification mechanism that can then be used to trigger events for mitigating management actions. When violation event occurs, the violation information will be sent out to the subscribed party.

The functionality of the Control Loops applied to the Negotiation process refers to:

- Application Control: The requirements expressed in the Application-SLA allow for negotiation of SLAs and as a result reservation of resources according the application's QoS requirements. Changes in the requirements are reflected in the Application-SLA and as a result in the selected resources.
- Environment Control: The resource specification descriptions that are being produced by the Performance Estimation Service turn the Application-SLA requirements into Technical-SLA requirements through the mapping of high-level terms to low-level resource estimates. Actual application and resource behavior is reported by a feedback loop, this is used to check whether the required QoS is actually supplied by the IaaS provider. This data is also used to validate the accuracy of models used by the engine by comparing the predictions with actual measurements.
- Virtualization Control: The Technical-SLA commitments allow for resource provisioning according to the application requirements expressed. Given that a pre-reservation takes place while creating the offers according to the resource specification descriptions, the virtual resources provided meet the application's QoS. Changes in the infrastructure with regard to Technical-SLA commitments result in a domain wide resource availability check and a pre-reservation in the IaaS resources for computing, storage and network resources.

C. Reservation

The goal of the Reservation process is to reserve the resources (virtual and physical) according to the agreed Technical-SLA between the PaaS and the IaaS provider. This process is part of the negotiation process described above. The main responsible actor is a Deployment Manager that is

instantiated by the ISONI SLA Manager. The tools / services engaged in this process are the following (additional information on these services can be found in [8], [9]):

1) Deployment Manager: A service responsible for the deployment of specific virtual networks in the IaaS.

2) Resource Manager: A service responsible for managing the execution resources (compute & storage) within the IaaS domain.

3) Storage Manager: A service responsible for managing the reservation of storage resources within the IaaS domain.

4) Path Manager: A service responsible for managing the network resources within the IaaS domain.

The functionality of the Control Loops applied to the Reservation process refers to:

- Environment Control: The Technical-SLA reservations are communicated by the IaaS providers to the PaaS providers. Given that the Workflow Enactor Service resides on the PaaS, while an instance of it is also deployed in the virtualized environment, any change to the configuration from the Application Control Loop results in changes in the reservation through the Workflow Enactor Instance.
- Virtualization Control: The physical resources are being reserved and allocated according to the *resource specification descriptions*, which are part of the Technical-SLA. These descriptions contain functional requirements for the deployment with respect to computing, storage and networking. When the virtual network needs to be instantiated the respective resources are brought into service via the Virtualization Control interface. In order to set up or configure resources, for computing and storage the Resource Manager instructs the Execution Environment (EE) and for networking the Path Manager instructs the ISONI eXchange Box (IXB) [10].

V. PROCESSES GUARANTEEING REAL-TIME

The goal of the channels / processes described in this section is to provide the necessary functionality to guarantee QoS during execution. Therefore, we describe (besides execution) two main processes: Monitoring and Re-negotiation. The first one is a fundamental process that allows for evaluation of metrics during runtime in order to ensure that the reserved resources meet the application requirements, while the second one may either be triggered by the application at runtime (e.g. more users in a collaborative session) or by the IaaS providers if the initially expressed application requirements cannot be fulfilled with the reserved resources and re-negotiation is needed in order to guarantee the QoS.

A. Execution and Monitoring

The goal of the Execution and Monitoring process is to enable execution of the application according to the QoS requirements, while monitor allows measuring QoS at both application and infrastructure levels targeting trigger events for runtime adaptability of resource provisioning estimation and decision making. The main actors in this process are the SaaS Provider, the PaaS Provider and the IaaS Provider. The tools / services engaged in this process are the following:

1) Deployment Manager: During the execution phase, inside the IaaS provider, the Deployment Manager is responsible for collecting infrastructure monitoring information and sending this low level information to the Monitoring Service (inside the PaaS provider) in the configured form requested during the reservation (Technical-SLA).

2) Resource Manager: During the execution phase the Resource Manager is responsible for collating and forwarding monitoring information received from the infrastructure regarding the resources.

3) Path Manager: During the execution phase the Path Manager is responsible for collating and forwarding monitoring information received from the infrastructure regarding the network links.

4) Workflow Enactor Service: A service [11], inside the PaaS provider, responsible for configuring, starting and stopping the applications (an instance of the service is deployed in the virtualized environment to invoke the services).

5) Monitoring Service: A service [12], inside the PaaS provider, responsible for collecting the high as well as the low level information provided (an instance of the service is deployed in the virtualized environment to monitor the ASCs and provide corresponding reports to the monitoring service).

6) Storage Manager: During execution phase the Storage Manager is responsible for collating and forwarding monitoring information received from the infrastructure regarding the storage units.

7) Real-time Scheduler: A service allowing for temporal isolation among concurrently running VMUs, in such a way that the temporal interferences among them do not disrupt the QoS guarantees required by the applications running within the VMUs [13]. This mechanism provides strong scheduling guarantees for an entire VMU, since it ensures a configurable CPU time within a guaranteed repeating maximum period of time. The application component running inside a VMU benefits from these real-time guarantees by experiencing a constant CPU performance as if it were running alone on the physical system. The advantage of this scheduler over the priority-based ones is the ability to provide temporal encapsulation among competing processes, ensuring that an individual process inside the VMU runs with proper QoS guarantees.

8) ISONI eXchange Box: A service [10] regulating concurrent deployments regarding networking resources in order to manage and guarantee the bandwidth. Flow control ensures that the virtual networks are really isolated and do not impact each other.

9) Storage QoS Manager: Based on storage pool's and associated VMU connections' QoS parameters the Storage QoS Manager enforces storage quality of service on each VMU connection.

10) Execution Environment: The Execution Environment is a framework in which a VMU is running. It also adds additional features including real-time enabled execution through the Real-time Scheduler and features for redundancy, migration and the connection to the long term storage. It also provides an endpoint for the connection to the virtualized network for the interaction between different VMUs.

The functionality of the Control Loops applied to the Execution process refers to:

- Application Control: The information provided by the application monitoring allows for application QoS provision, since it contains critical ASC outputs (e.g. fps of a teleconference, response time of a server etc.). This runtime information is relayed to the Monitoring Service in order to be utilized by the PaaS provider for taking corrective actions that will ensure that the real-time guarantees are kept throughout application execution.
- Environment Control: The Application-SLA and Technical-SLA metrics are being monitored during the execution since monitoring information is collected both from the applications and from the infrastructure. Violations are communicated to the SaaS and IaaS provider in order to take corrective actions. This may result to SLA Re-negotiation (as explained in the following section of this paper).
- Virtualization Control: The physical resources are monitored during execution. Any violation that cannot be handled within the IaaS domain (e.g. through live migration) is reported to the SLA Manager and escalated as a T-SLA violation.

B. Re-negotiation

The goal of the Re-negotiation process is to provide updated resources at runtime following either a request from a Customer or the monitoring information (obtained during execution) that shows that the initially expressed application requirements cannot be fulfilled with the reserved resources and re-negotiation is needed in order to guarantee the QoS. The changes affect the virtual network and refer to: Computational power, Memory, Bandwidth, Storage, and Lifetime of virtual networkds. Re-negotiation may be triggered during execution, which actually means that all artifacts and components are in place. As in the Negotiation process, the actors in this process are: the Customer, the SaaS Provider, the PaaS Provider and the IaaS Provider.

The functionality of the Control Loops applied to the Re-negotiation process refers to:

- Application Control: The application configuration is changed at runtime which is reflected to the configuration information that is passed to the Environment Control Loop through the Workflow Enactor Service.
- Environment Control: The updated *resource specification descriptions* that are being produced by the Performance Estimation Service include the updated resource estimates. The latter allows for real-time handling of QoS exceptions and changes in uncertainty throughout the application execution.
- Virtualization Control: The updated resource allocations are based on the new descriptions as part of an updated Technical-SLA. The Deployment Manager changes the virtual network according to these new descriptions.

VI. USE CASES

In order to demonstrate the difference between the various kinds of applications, interactive or non-interactive ones, and their according needs we have performed a number of experiments. First, for a non-interactive application, which consists of a Matlab benchmark test (mixed integer and floating point operations), the results (**Error! Reference source not found.**4 and **Error! Reference source not found.**5) are very close to the theoretical expectance. Due to the fact that this process runs as standalone and no interactivity is required, the overall score which depicts the performance is linear to the amount of resources assigned to it (CPU %). Furthermore, this assignment is not dependent on the underlying scheduling parameters like the budget C and the period D over which the percentage is assigned.

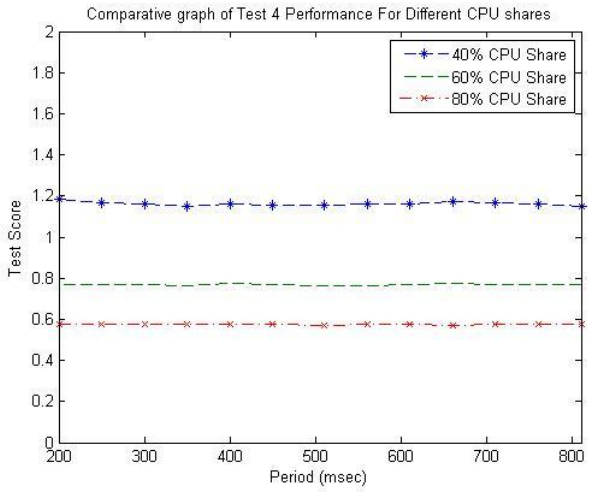

Figure 4. Non-interactive test score performance for varying CPU shares and scheduling periods

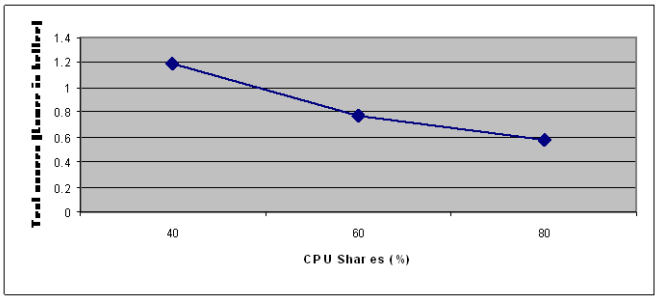

Figure 5. Test score in comparison to CPU shares

It is imperative to stress that the aforementioned graph was created after running the test score for about 500 seconds for

each configuration and taking the mean of the performance in all the runs that were conducted in this interval. This resulted in hundreds of executions for each configuration.

On the other hand, for an interactive application such as an application server, whose main task is to receive incoming requests, search through a database and produce the result, another experiment was conducted in order to observe the effect of the scheduling parameters and mainly of the period P over which a specific CPU share is assigned. As it appears in **Error! Reference source not found.**6 and **Error! Reference source not found.**7, while the mean response time of the server for each configuration is not significantly affected by the scheduling parameters, probably due to the fact that the computational time needed to process each request is the same, this is not the same for the standard deviation of the response times. For that metric, choosing a proper scheduling period is critical, given that a request that arrives just before the end of the activation period of the task will have to wait until the eminent deactivation period is finished in order to be processed. Thus the poor selection of a scheduling period will lead to a service that is for specific times very fast and for other cases completely unresponsive. However, a constant level of QoS is imperative in order to ensure a smoothly running interactive application. From the measurements it is evident that having a low scheduling period is very helpful for maintaining that stable QoS level. The latter is achieved in the Execution and Monitoring process that allows for provision of real-time guarantees through the real-time scheduler.

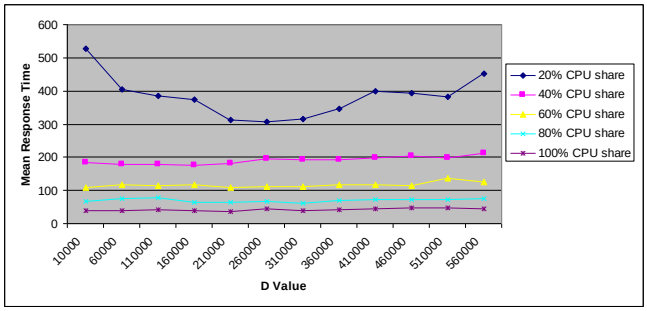

Figure 6. Mean value of response times for different CPU shares and periods

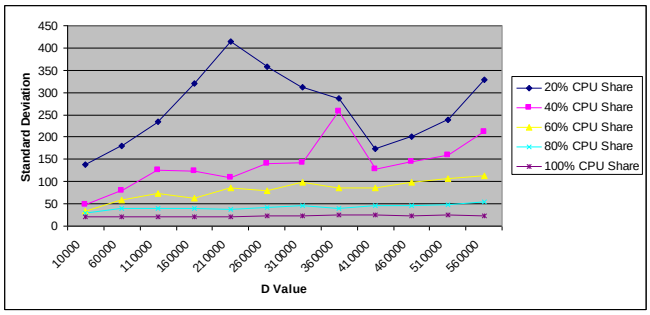

Figure 7. Standard deviation of response times for different CPU shares and periods

VII. CONCLUSIONS

Current approaches on service oriented architectures focus on designing and implementing a rich set of services to efficiently operate, manage and reconfigure computing, storage and network resources under real-time conditions, providing to end users and to the associated applications the appropriate and required level of QoS. All Platform and Infrastructure capabilities are offered as on-demand services, although the architecture of the media applications varies from traditional n-tier enterprise applications to service-oriented workflows. Thus emerging cloud-based platforms and service oriented infrastructures face the challenge of providing QoS guarantees in order to facilitate real-time and interactivity as requested by Future Internet Applications.

In this paper we described how real-time aspects are addressed across all layers of service oriented environments. The proposed approach and the architecture has been developed within the framework of the IRMOS project; being validated with three different real-time interactive multimedia applications, namely Digital Film Postproduction, Interactive Real-time eLearning and Virtual and Augmented Reality. We have also briefly demonstrated the need for a real-time scheduler in one of the processes to guarantee real-time, the Execution and Monitoring process.

REFERENCES

[1] T. Erl, “Service-oriented Architecture: Concepts, Technology, and Design”, Upper Saddle River: Prentice Hall PTR, ISBN 0-13-185858-0, 2005.

[2] The NIST Definition of Cloud Computing, Peter Mell and Tim Grance, Version 15, http://csrc.nist.gov/groups/SNS/cloud-computing, 2009

[3] The IRMOS Project, www.irmosproject.eu

[4] IRMOS Whitepaper, “Intelligent Service Oriented Network Infrastructure Whitepaper”, 2009.

[5] Boniface, M., Nasser, B., Papay, J., Phillips, S., Servin, A., Zlatev, Z., Yang, K. X., Katsaros, G., Konstanteli, K., Kousiouris, G., Menychtas, A., Kyriazis, D. and Gogouvitis, S., “Platform-as-a-Service Architecture for Real-time Quality of Service Management in Clouds”, Fifth International Conference on Internet and Web Applications and Services, ICIW 2010, May 2010, Barcelona

[6] Kyriazis D, Einhorn R, Furst L, Braitmaier M, Lamp D, Konstanteli K, Kousiouris G, Menychtas A, Oliveros E, Loughran N, Nasser B, “A Methodology for engineering real-time interactive multimedia applications on Service Oriented Infrastructures”, IADIS Applied Computing 2010, Timisora, Romania, 2010

[7] IRMOS Project Deliverable D3.1.3, "Updated Version of IRMOS Overall Architecture ", 2009

[8] Oberle K, Kessler M, Voith T, Stein M, Lamp D, Berger S, “Network Virtualization: The missing piece”,ICIN2009, Bordeaux, 2009

[9] Oberle K, Voith T, Stein M, Gallizo G, Kübert R, “The Network Aspect of Infrastructure-as-a-Service”, ICIN2010, Berlin, October 2010

[10] Kessler, M., Oberle, K., Braun, S., Lamp, D., “Network Virtualization: Towards a fully virtualized service infrastructure”, ISC2009, Hamburg, 23-26.06.2009

[11] Gogouvitis S, Konstanteli K, Kousiouris G, Katsaros G, Kyriazis D, Varvarigou T, “Workflow Management in Service Oriented Infrastructures”, Proceedings of the 6th IEEE/IFIP International Conference on Network and Service Management, Canada, 2010

[12] Katsaros G, Kousiouris G, Gogouvitis S, Kyriazis D, Varvarigou T, “A Service Oriented Monitoring Framework for soft real-time applications”, IEEE International Conference on Service-Oriented Computing and Applications (SOCA), Australia, 2010

[13] Checconi F, Cucinotta T, Faggioli D, Lipari G, "Hierarchical Multiprocessor CPU Reservations for the Linux Kernel," 5th International Workshop on Operating Systems Platforms for Embedded Real-Time Applications (OSPERT 2009), Dublin, Ireland, June 2009

[14] Kousiouris G, Kyriazis D, Konstanteli K, Gogouvitis S, Katsaros G, Varvarigou T, “A Service-Oriented Framework for GNU Octave-Based Performance Prediction”, to appear in Proceedings of the IEEE International

Conference on Services Computing Conference (SCC), Miami, USA, July 2010.

[15] Cucinotta T, Checconi F, Kousiouris G, Kyriazis D, Varvarigou T, Mazzetti A, Zlatev Z, Papay J, Boniface M, Berger S, Lamp D, Voith T, Stein S, "Virtualised e-Learning with Real-Time Guarantees on the IRMOS Platform," to appear in Proceedings of the IEEE International Conference on Service-Oriented Computing and Applications (SOCA 2010), Perth, Australia, December 2010

[16] G. Gallizo, R. Kübert, Karsten Oberle, Klaus Satzke, E. Oliveros, S. Gogouvitis, G. Katsaros, "A Service Level Agreement Management Framework for Real-time Applications in Cloud Computing Environments", CloudComp 2010, Barcelona, Spain, 25.10-28.10.2010

Dimosthenis Kyriazis received the diploma from the Dept. of Electrical and Computer Engineering of the National Technical University of Athens, in 2001, the MSc in "Techno-Economic Systems" in 2004 and his PhD from NTUA in 2007. He is currently a Research Engineer in the Telecommunication Laboratory of the Institute of Communication and Computer Systems (ICCS). Before joining the ICCS he has worked in the private sector as Telecom Software Engineer. He has participated in numerous EU / National funded projects (such as NextGRID, Akogrimo, BEinGRID, HPC-Europa, etc) and currently serves as the Technical Coordinator of the IRMOS project. His research interests include scheduling, Quality of Service and workflow management in heterogeneous systems and service oriented architectures.

George Kousiouris received his diploma in Electrical and Computer Engineering from the University of Patras, Greece in 2005. He is currently pursuing his PhD in Grid Computing at the Dept. of Electrical and Computer Engineering of the National Technical University of Athens and is a researcher for the Institute of Communications and Computer Systems, where he has participated in the EU funded projects BEinGRID, IRMOS, Challengers and the National project GRID-APP. In the past he has worked for private telecommunications companies and the Hellenic Air Force. His interests are mainly computational intelligence, optimization, computer networks and web services discovery.

Thomas Voith is a senior scientist in Bell Labs Services Infrastructure Domain in Stuttgart, Germany. He received his degree in Electrical Engineering from the University of Stuttgart. Currently he is working on disruptive technologies – related to infrastructure as a service – that provide a superior foundation for Alcatel-Lucent's application enablement strategy. Mr. Voith worked on innovative architecture designs for next generation infrastructure in conjunction with European and national German research programs. He was an active member and editor of International Telecommunication Union, Telecommunication Standardization Sector Study Group 11 for several years. He joined the European Telecommunications Standards Institute – Telecommunication and Internet converged Services and Protocols for Advanced Networks during preparation of Next-Generation Network Release 1.

Eduardo Oliveros Diaz holds a Telecommunication Engineer Degree from Universidad Politecnica de Madrid (1997). He worked for Future Space (1998) where he participated in the project Infomail (Multimedia Messaging Server) of Telefonica R&D. He joined Telefonica R&D in 2000, in the "Messaging Services" division, participating in the development of different e-mail Systems, Directory and Voice Portals solutions for Spain and South America. Following he joined the "Real-time communications services" division where he was involved in several Telefonica Corporation innovation projects concerning Directory and Unified Messaging Systems. He has participated in the European project Akogrimo involved in exploitation activities and in the Administrative Coordination in the European project BREIN. Currently he is working in the IRMOS project.

Andreas Menychtas graduated from the School of Electrical and Computer Engineering, National Technical University of Athens (NTUA) in 2004. In 2009, he received his PhD in area of Distributed Computing from the School of Electrical and Computer Engineering of the National Technical University of Athens. He worked in the private sector as computer and network engineer and has been involved in several EU and National funded projects such as NextGRID, EGEE, IRMOS and GRID-APP. His research interests include Distributed Systems, Web Services, Object Oriented Programming, Service Oriented Architectures. Currently, he works as research engineer in the Institute of Communication and Computer Systems of NTUA.

Karsten Oberle received the degree in Communications Engineering from the University of Applied Sciences, Mannheim, Germany, in 1998. In the same year he joined the Alcatel Research Center in Stuttgart. Currently he is Project Manager in the Bell Labs Service Infrastructure research department of Alcatel-Lucent Germany. The Service Infrastructure research focuses on disruptive technologies that provide a superior foundation for Alcatel-Lucent's application enablement strategy. In 2008 he has been awarded for his technical excellence becoming member of the Alcatel-Lucent Technical Academy. He has participated in numerous national and European projects. Karsten is active in Standardization, while since 2008 he holds the position of the Vice-Chairman of ETSI CLOUD. Since 2007 he has authored and co-authored more than 20 papers for conferences and journals. Furthermore he has filed more than 20 patents.

Michael Boniface is Technical Director of the IT Innovation Centre. He joined IT Innovation in 2000 after several years at Nortel Networks. His roles at IT Innovation include technical strategy of RTD, technical leadership and business development. He has over 10 years experience of RTD into distributed systems for science and industry using technologies such as Semantic Web, Grid and service-oriented architectures. He leads IT Innovation's contribution to the Future Internet initiative through Expert Groups including leadership of the socio-economic working group. Michael provides architecture and business modelling direction including scientific leadership of SIMDAT, leadership of sustainability activities in BonFIRE and FIRESTATION, socio-economic impact assessment in SESERV and he provides overall coordination for the GRIA software.

Tommaso Cucinotta graduated in Computer Engineering at the University of Pisa in 2000, and received the PhD degree from Scuola Superiore Sant'Anna in 2004. He is assistant professor at the real time systems laboratory of scuola superiore sant'Anna in Pisa. His main research activities are in the areas of real-time and embedded systems, with a particular focus on real-time support for general purpose Operating Systems and virtualised real-time applications.

Sören Berger reached his Dipl.-Inf. degree at the University of Stuttgart in 2008 where he was working on open source VoIP campus solutions and at robustness and reliability in future IT infrastructures and distributed systems. Now he works at the Rechenzentrum Universität Stuttgart (RUS) at the University of Stuttgart and carry out research on resource management in distributed systems and visualization of network and computing resources in cloud infrastructures.