Interactive Social TV on Service Oriented Environments: Challenges and Enablers

Dimosthenis Kyriazis, George Kousiouris, Andreas Menychtas, Anastasios Doulamis, Theodora Varvarigou
National Technical University of Athens
Athens, Greece
{dimos, gkousiou, ameny}@mail.ntua.gr, adoulam@cs.ntua.gr, dora@telecom.ntua.gr

Abstract — Future Internet Applications require the on-demand provision of ICT assets with regard to data, services and process technologies. What is missing from today’s service platforms to realize this vision refers to the way services are perceived to work and as a result being composed and managed. In this paper we present an application scenario, namely Interactive Social TV that highlights the need to separate the concepts of functionality, content and context within the services domain in order to enable stream and event handling. Furthermore and given the need to meet the requirements of future internet applications, we introduce the concept of functional programming models as a means to allow goal-oriented composition of media-rich services with specific QoS requirements.

Keywords – interactive multimedia; social TV; service oriented architecture;

I. INTRODUCTION

Today, service platforms realize virtualization technologies in order to support and enable the provision of different kinds of resources with guaranteed Quality of Service (QoS) levels. However, current approaches mainly focus on the virtualization of networking links, storage and computational nodes, striving to increase the maximum achievable system utilization. Moreover, one has to take into consideration that in the years to come it is expected that anything that is able to provide an output (commonly referred to as: “streaming data”) will be integrated in a network in order to take advantage of the information delivered. In this context, future emerging applications, such as Social TV, will not only use but also pose specific requirements to the service platforms, which results to an increased demand of services and an increased complexity of their composition and management.

The aforementioned service platforms are mainly implemented following the principles of Service Oriented Architectures (SOAs) [1], which 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 four 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.
- Context as a Service (CaaS), which refers to the provision of contextual information relevant to the service consumption and delivery in order to optimize resource usage and maximize the user experience. For example, users of mobile telephone networks might send their location, billing profile, etc.
- Software as a Service (SaaS), which refers to the provision of an application as a service over the Internet or distributed environment.

While work has been carried out in an attempt to offer service-based platforms in the context of these cloud classes, e.g. the IRMOS project [3] offers a SOI able to support interactive real-time multimedia applications [4], there are still many challenges to face with regard to future internet applications that pose specific needs for interactivity, data management, event handling, etc.

Service convergence is expected to reshape sectors such as media, broadcasting and telecommunications. Products will be offered as services, many of which will be media-rich, but also storage, computation and network intensive and cost sensitive. To this direction, new service platforms should enable the provision of content and media as a service [16] through an intelligent representation and handling of streams and events. Such platforms besides enabling stream processing and synchronization, should also be able to address issues related to information extraction from the streams. This includes the definition of the context, triggering and accumulating events based on the contextual information in addition to reacting in an automated way based on event analysis.

In this paper we discuss the challenges as well as initial approaches on how these challenges can be addressed in order to enable the execution of a virtual environment on a service
The remainder of the paper is structured as follows: Section II presents a sample scenario - interactive broadcasting media, in order to depict the gaps in today’s service platforms (analyzed in detail in Section III - namely “Challenges”) and highlight the challenges and the enablers (discussed in Section IV) of such a scenario. The paper concludes with a discussion on future research and potentials for the current study.

II. SCENARIO DESCRIPTION

The interactive broadcasting social TV scenario includes two (2) end-users that act both as viewers of video streams but also as producers of streams in an interactive session. Based on that, each of the users views through the display equipment two (2) streams: the video that is being broadcasted from a broadcasting centre as well as the video of the other viewer. They both see the same video stream at the same time while they also see each other in the same display (the video streams are mixed) as it would have been if they were sitting together to watch the video (e.g. movie). Furthermore, the communication between the two viewers is enhanced with additional services such as instant messaging......

Figure 1. Interactive Broadcasting Media / Social TV

The sequence of events that is depicted in the above figure is described as follows (exceptional cases are also discussed in order to highlight the needs towards future service platforms):

1) A Broadcasting Centre is providing a video stream (e.g. movie) that is displayed in both Viewers at the same time.

2) (Exceptional Case) The processing capabilities of the Viewer 1 Display are not adequate to support real-time simultaneous stream processing coming from two different sources.

3) (Exceptional Case) While transmission takes place, there is a failure in the host of the service that streams the video from the broadcasting centre.

4) Viewer 1 has a stereoscopic camera while Viewer 2 doesn’t have one - she / he has two (2) cameras. However, Viewer 1 would like to receive a 3D video stream from Viewer 2 (who doesn’t have a stereoscopic camera).

5) Stream 1 is produced by the camera of Viewer 1 while Stream 2 is produced by the camera of Viewer 2. These video streams are transmitted to a Stream Processing service that mixes the streams with Stream 3 and produces a new mixed Stream – Stream 4 – forwarded to the displays of Viewer 2 and Viewer 1 correspondingly. Each viewer would like to see the other viewer at the same time (synchronized).

6) Stream 1 and Stream 2 are being processed by platform service (Context Definition) that define whether the viewers enjoy the content of the video (e.g. laughing) or not. While Viewer 1 smiles, Viewer 2 doesn’t seem to enjoy the movie. Two (2) events - Event 1 and Event 2 - are triggered from the Context Definition service in order to inform the Broadcasting Centre that one of the viewers doesn’t enjoy the movie.

7) Event 1 and Event 2 are being processed by the Event Handling service which accumulates the events and triggers another event - Event 3 - to the Broadcasting Centre.

8) (Exceptional Case) Two events happen at the same time: a. the processing capabilities of the Viewer 1 Display are not adequate to support real-time simultaneous stream processing coming from two different sources; b. While transmission takes place, there is a failure in the host of the service that streams the video from the broadcasting centre.

9) Broadcasting Centre analyzes Event 3 and transmits a new video stream - Stream 3 - to the Stream Processing service that mixes it with Stream 1 and Stream 2 and transmits it to both of the viewers.

10) (Exceptional Case) A new Viewer enters the session but without a camera. Viewer 3 has only a microphone and a display though which she watches the same video clip with Viewer 1 and Viewer 2.

11) (Exceptional Case) Viewer 1 and Viewer 2 decide to switch their application from video to an audio clip.

III. CHALLENGES

While significant advancements have been presented in the related fields during the last years, the adoption of virtual environments on service oriented infrastructures is in its early days. Based on the sequence of events presented in Section II, a number of challenges arise that are not sufficiently met from today’s platforms.

Challenge 1: Context definition and rich representation of streams and events to allow optimum handling of them. This challenge is linked with issues “6”, “7” and “9” of the scenario. The platform must be in a position to understand viewer reactions to the presented content and make according decisions regarding the provided content. Currently context definition is performed statically, and mainly through semantic annotation of data and is used only at the beginning of the process, with no run-time feedback from the user and automatic adjustment from the platform. Streams, events and in general any data (either on the application or on the system
level) has to be represented in a way that will allow for the definition of context out of data and thus triggering of events based on the aforementioned context.

Challenge 2: Resource abstraction and modeling for automatic identification of resource demand at runtime. This challenge is linked with issues “2” and “11” of the scenario. Prior to any interaction with the available resources, these need to be profiled and modeled in a complete and abstract way so as to enable the capabilities of infrastructure to deal with them in an automated way during runtime. While this challenge is addressed with today’s approaches for many hardware devices (e.g. computing and storage nodes), a common representation for all resources (e.g. displays) is missing and therefore the capabilities of such resources are not known and cannot be either exploited or handled accordingly by current platforms. For example, when the application is altered (from joint video viewing to simple audio) the need for resources is also altered and additionally the different types of applications may have different critical operational parameters such as jitter or delay. However, in most of the distributed systems infrastructures the resource reservation dynamicity during runtime is limited and the resources allocated to the user can be only decided and reserved a priori, during the SLA negotiation phase.

Challenge 3: Infrastructure-enabled resources and full incorporation of them and their functionality. This challenge is linked with issues “2” and “4” of the scenario. When a resource is attached to the infrastructure (like the cameras mentioned in the above scenario), the latter must have a notion of its capabilities and functional framework (e.g. if it contains storage or processing capabilities, what formats it supports etc). If this resource does not have sufficient capabilities to meet the application requirements, the platform must provide the means to overcome this problem. Up to now, this process is conducted manually, with the use of semantic technologies like ontologies. However, this implies that the application user or the manufacturer must have described the device, while the same concepts must exist over all manufacturers (they all have to follow the same ontology definition). Moreover, resources with limited capabilities (e.g. non-stereoscopic cameras) are not exploited with today’s platforms - though these resources could be fully exploited by providing their “missing” capabilities through the infrastructure.

Challenge 4: Functionality-based services enabling automated service composition. This challenge is linked with issues “4” and “10” of the scenario. Description: If two resources or services are not compatible, then their communication is not feasible. There is no automated transformation from one format to the other, or automatic combination of existing services in order to extend their abilities. Even with the Web Services, which guarantee cross-platform compatibility, there must be a manual interconnection between the interfaces. Moreover, there are cases where a new service is required to handle with the new type of data (e.g. the new user in the scenario only uses a microphone and not a camera) while the users and the providers of the information remain the same. With today’s approaches the service composition should change in order to include the new service and communicate the aforementioned change to the execution engine to re-initiate the services and the whole process. Functionality-based services are required in order to update the service composition with the new service in an automated way.

Challenge 5: Early prediction and proactive actions for both infrastructure and application failures / errors. This challenge is linked with issues “3” and “8” of the scenario. During runtime, a number of parameters such as degrading network or failing hardware may impose difficulties in the operation of the platform. Current solutions are mainly based in infrastructure messages / notifications towards the application or the platform, communicating failures and errors. However this information usually arrives too late or even possibly after the failure, in order to determine what part of the chain failed. Moreover, today’s approaches only handle infrastructure errors while application issues are dealt from the application with the deployment usually of monitoring mechanisms and predefined policies for reactions to errors as soon as they appear. There is not an automated way of handling such errors in a complete manner. To this direction, event patterns’ analysis is required in order to allow identification, tracking, prioritization and prediction of events in a non-deterministic way. The above highlight that there are two kinds of events: a) related to the content / context and b) related to the infrastructure. Both of these not only have to be handled in future service platforms but also used to predict possible failures and trigger mechanisms that will proactively deal with them.

Challenge 6: Adjusting execution management based on infrastructure capabilities and application demands. This challenge is linked with issues “5” and “11” of the scenario. During both the initialization of the process but also at runtime based on the stream and event modeling services, the platform should adjust the execution management following the infrastructure capabilities (e.g. move mixing of streams close to source or close to the synchronization point according to network links, processing capabilities of nodes, etc) and application demands (e.g. synchronization may require caching of content at a specific point in the infrastructure). In addition, when the application changes (as in the scenario to an audio-clip) the infrastructure requirements change and execution management should be adjusted according to the new requirements and scale-down for example.

IV. ENABLERS

In order to enable all the aforementioned functionalities and be able to implement the scenario, service oriented environments should be able to facilitate specific mechanisms / frameworks, which are being discussed in this section.

A. Stream & Event Modeling

Through context definition, key features of the service or the application may be investigated and their requirements and according configuration of the rest of the infrastructures may be achieved. Services will be data-oriented allowing for the definition of context out of data, which will be linked with the semantic definition of a resource and the according functional provisioning in the service level. Moreover, through event recognition from both sources (application content and infrastructure status) resource management can be performed in a proactive and optimized manner. To this direction, a
Unified Data Model is required in order to allow representation of streams, events and any kind of data (both on application and on the system level). This model should be rich enough to capture relationships and explore the interactions between functionality, context and data. Related work, like the ones found in [8], [9], and [10], borrowed from the field of computer vision and image processing may prove extremely helpful in this attempt to process content created by the user (her/his stream) and deduct conclusions regarding her/his satisfaction (as identified in Steps 6 and 7). This way, Challenge 1 may be met.

B. Resource Aspects

Due to service platforms’ consideration for resource functionality, intelligent decisions should be taken regarding infrastructure management. With the ability to incorporate devices like cameras in the above scenario and evaluate their capabilities, special support should be given for sources and sinks, based on the analysis of streams and the identification of events. This can be interpreted as a way to adapt the service deployment according to the infrastructure capabilities (which include not only traditional IT resources but also other types of resources - e.g. cameras) and the nature of data that these devices can handle. With the use of semantic-rich patterns and rules to describe devices and attach contextual information (e.g. device properties, location, time, user preferences, etc), their abstraction and management, as resources that can be shared and combined will be enabled.

Based on the above, a Resource Model is essential in order to allow for a common representation of resources through the abstraction and profiling of them; thus enabling the specification of their characteristics with respect to the physical limitations they impose when becoming part of a virtualized infrastructure (e.g. device performance limitations, location and as a result network links to cameras, etc). The standards and semantics related to resource types should be extended in order to raise the abstraction level and be able to handle not only today’s infrastructure resources (such as processing, storage and network) but also other resource types. Furthermore, profiling of the resource will allow for their repurposing (e.g. cameras that are used under a specific procedure may be repurposed and used for a different one, which also applies in the case a camera is not working properly) extending the pool of available resources in the infrastructure and their utilization. An interesting work in this domain appears in [11].

C. Service Aspects

From a service aspect, the main issue regards orientation towards functionality-based services that will be able to exploit the abstracted resources. Principles of functional programming should be followed in order for the implemented services to be more function-oriented, more generic and in order to allow for automated service orchestration or adaptation. This will lead to real world devices being optimally integrated in the infrastructure, especially in the media domain, as this is foreseen in the specific scenario.

Furthermore, adaptation of functional services will lead to a more generic platform that will be able to self-combine different infrastructures in order to achieve added value from this action and lead to new applications more adapted to the physical layer. The separation of service concepts (functionality, context and content) will shift the current service orientation (request / response) to content and functionality orientation (publish subscribe model). The latter is of major importance since new services may be created automatically with specific functions (core shared functionalities that can be used to deliver, process, filter and aggregate information); a process non-transparent to the operational management “layer” since the services will only expose their functionality and an API.

The platform services refer both to the ones for management of the infrastructure (e.g. Service Level Agreement negotiation, accounting, selection, etc) but also to the stream and event aware ones (e.g. event prediction). Through stream processing, information may be inferred regarding the type of the needed service (e.g. what is the core requirement, accuracy of info, delay etc.). This type of event and stream processing will make infrastructure control more efficient and adaptable, enabling features such as dynamic handling of resources and achieving a better trade-off between meeting QoS requirements and utilization. In state of the art approaches regarding multimedia applications and SOIs (like in [3]), this is done mainly through a specified interface between the applications and the platform, which inserts an extra layer of complexity and application adaptation that could be avoided with the aforementioned proposal.

Hardware profiling [17] should be enabled in order for the platform to be able to adapt the data streams to the former or vice versa, from selecting for example which tool to use or which mode to implement for the communication or where the main processing will take place. Based on what is defined in the standards, service environments should expose media processing algorithms as services for information extraction and synchronization of streams and going beyond the state of the art by taking into account the contextual information which results to different data rates, confidence levels, granularity levels and properties related to the environment. Moreover, methods and tools for handling event generation, prioritization (to identify and describe the severity of each event and detecting failures of critical system elements) and different ways of event processing including automatic reaction should be provided. The latter is of major importance since early detection of serial errors will be recognized from an accumulation of corresponding events avoiding resulting errors. Event processing should be based on patterns analysis including identification of patterns’ appliance consequences that produce context and which can be possibly resolved by other patterns and asynchronous notification of events.

D. Operational Management

Through functional programming’s principle of “function as input to functions”, service orchestration can be conducted in more advanced level with greater combinations and abstraction. Due to the stateless nature of this process (we are not interested in past states, the key aspect is the reaction of the platform to the current stream and according event) scalability is greatly enhanced and parallelization of applications is feasible. By combining functionality-based services and
semantics of resources, automated composition of services may be performed. Final model checking is feasible via investigation of the data flow in the application, a strong feature that is supported by Functional Programming. Through features such as automatic data type conversion, the bridging between two devices may be performed, a feature that currently requires specific interventions from the providers. The same applies for key functionalities of this programming paradigm such as the usage of different functions with the same arguments. Applying the functional programming model to operational management and allowing implementation based on functionality will enable switching between services with the same functionality but different interface characteristics due to automatic type conversion. Moreover, it provides the ability to use the same service instance with different input types (through type inference - “polymorphism”) and it provides scalability on the service level due to the inheritance concept.

Functionality-based services [18] will enable automated service composition through functional programming principles (e.g. “polymorphism”). Since higher abstraction is feasible (services can be submitted as input to services) through data type conversions or usage of the same services with different arguments, while at the same time resource combination is feasible since no specific composition is needed for this purpose. Besides, the type checking concept should be applied to check whether two services can cooperate or not and whether the data flow and transformation within a workflow are possible. In the figure that follows we depict how service composition can be simplified and automated through functional programming given that there will not be a need for different services based on input (e.g. service for filtering a video clip - f1 - is different from a service filtering an audio clip - f2) but composition will be done with regard to functionality (in the aforementioned example: filter). This would be a significant add-on to worthwhile existing approaches for service composition like the ones presented in [12], [13], [14] and [15]. Especially for [13], this also supports the publish/subscribe model identified in sub-section – Service Aspects.

Finally, contextual information should also be used to adjust operational management in order to optimize content delivery given the resource model. Computation may move either close to the source (if for example the network links attached to it are not adequate for the streams synchronization) or close to the synchronization point in case the resources that are close to the users are not capable enough to perform stream processing. The trade-offs will also be studied with regard to the cost in each different case.

V. CONCLUSIONS

Current approaches on service oriented infrastructures focus on designing and implementing a rich set of services to efficiently operate, manage and reconfigure computing, storage and network resources under various 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. However, future internet applications mainly refer to virtual worlds and environments, which pose specific needs towards service based platforms.

In this paper we presented the challenges and proposed approaches to address specific topics related to these challenges, when dealing with virtual environments in service oriented environments. We have discussed an application scenario, namely interactive social TV, as a means to depict the requirements of virtual environments in terms of interactivity, hardware resources that are not limited to traditional IT ones (i.e. computing, storage, networking) but also refer to digital equipment (e.g. cameras), context definition, stream and event management. Furthermore, we proposed how these challenges can be addressed by service based platforms that seek to adopt future internet applications.

REFERENCES


