PCS: Persistent Collaboration Sessions in Multiscreen Environments

Sung-Soo Kim
ETRI
Daejeon, South Korea
sungsoo@etri.re.kr

Chunglae Cho
ETRI
Daejeon, South Korea
clcho@etri.re.kr

Jongho Won
ETRI
Daejeon, South Korea
jhwon@etri.re.kr

ABSTRACT

Multiscreen devices create new opportunities and challenges for collaboration services.

In this paper, we propose a mobile middleware for symmetric collaboration among associated mobile applications in the same network environment. This paper focuses on the challenge of providing the seamless services. The proposed middleware supports seamless collaboration services by maintaining the collaboration session even if the user changes one of the smart devices which participate in the collaboration session. The application developer can implement the collaboration-based multiscreen services easy and fast by using major functions in the middle ware API, such as remote execution, session join, session invitation, push migration and pull migration.

The major advantages of the collaboration middleware are providing communication transparency and seamless collaboration service delivery regardless of the changes of physical device configurations for collaboration.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Middleware, Mobile Computing

Keywords

ACM proceedings, LaTeX, text tagging

1. INTRODUCTION

In recent years, there has been an increased interest in smart applications. The main reason for this has been the realization that many diverse applications need a generic middleware for managing applications and their context information. In this paper, we describe the collaboration persistency.

Middleware 2014 Industry Track, December 8-12, Bordeaux, France

A Smart Home environment is a home equipped with sensors and activators of various types to monitor activities and movement, and to monitor risk situations, such as fire and smoke alarms.

The goal of mobile computing suggests including devices spanning the entire hardware spectrum. This augmentation includes the appliance of various use cases, including both the pervasive access to mobile services and ubiquitous communication between mobile hosts. Hence, those application cases can be reduced to the basic demand of communication among heterogenous devices in heterogenous environment. The ultimate goal of mobile middleware is to simplify the development of distributed applications. The goal of mobile middleware is to provide abstractions that reduce development effort, to offer programming paradigms that make developing powerful mobile applications easier, and to foster interoperability between applications. The essential role of middleware is to manage the complexity and heterogeneity of distributed infrastructures. On the one hand, middleware offers programming abstractions that hide some of the complexities of building a distributed application. On the other hand, there is a complex software infrastructure that implements these abstractions. Middleware is software that supports mediation between other software components, fostering interoperability between those components across heterogeneous platforms and varying resource levels. Mobile agents put the action where the data are, allowing programs to move autonomously about a network in order to access remote resources.

Service discovery middleware extends the client-server paradigm to include dynamic discovery of services and more dynamic interaction between clients and services. With service discovery middleware, developers can quickly develop highly dynamic client-server systems that are self-healing and support "plug and play" for individual components. The concepts embodied in service discovery architectures are not completely new; however, service discovery frameworks standardize the environments in which to deploy highly dynamic, self-healing client-server architectures.

In the fields of broadcasting especially IPTV and content delivery, multiscreen video describes video content transformed into multiple formats, bit rates and resolutions for display on smart devices such as television, mobile phone, tablet computer and computer.

The complexity of media will continually increase in terms of volume and functionality, thus introducing a need for simplicity and ease of use. Therefore, the massively distributed, integrated use of media will require replacing well-known interaction vehicles, such as remote control and menu driven search and control, with novel, more intuitive, and natural concepts.

Ambient intelligence aims to take the integration provided by ubiquitous computing one step further by realizing environments that are sensitive and responsive to the presence of people. The

focus is on users and their experiences from a consumer-electronics perspective.

The new paradigm aims to improve people's quality of life by creating the desired atmosphere and functionality through intelligent, personalized, interconnected systems and services. The term ambient refers to the environment and reflects the need for typical requirements such as distribution, ubiquity, and transparency. *Distribution* refers to noncentral systems control and computation. *Ubiquity* means the embedding is present everywhere. *Transparency* indicates that the surrounding systems are invisible and unobtrusive. *Intelligence* means the digital surrounding exhibit specific forms of social interaction. In other words, and environment must recognize the people that live in it, adapt itself to them, learn from their behavior, and possibly show emotion.

Key features of ambient intelligence: To refine the notion of ambient intelligence, Morzano et. al. formulated the following five key technology features. such as embedded, context-aware, personalized, adaptive and anticipatory.

In this paper, we propose a new mobile middleware to support the seamless collaboration services among heterogenous multiple mobile applications (or *apps*) in various smart devices. To provide convincing the collaboration services in mobile computing environments, we describe the key requirements of multiscreen service systems as follows:

- *Remote execution*: Latency is defined as the time between a player's action and the time the actual resulting game output on the player's screen. Since computer games are highly interactive, extremely low latency has to be achieved. The interaction delay in the games should be kept below 80ms in order to guarantee suitable user responsiveness.

- *Collaboration*: In order to provide a high-quality video (above 720p) interactively, data shall be reduced as much as possible but keeping quality. However, video encoding is computationally quite demanding.

- *Mobility*: In the case of network congestion, the network problems like increased latency, jitter, and packet losses distribute evenly on all competing traffic. However, the quality can be enhanced using quality of service (QoS) technologies to giver higher priority to game traffic in the network bottlenecks [**?**].

- *Synchronization*: Since the servers of cloud-based gaming service system have high-performance CPUs and GPUs, the operating costs for the servers are quite expensive. So, it is necessary to develop the optimization technologies to minimize power consumption and network bandwidth [**?**].

Motivation: In this paper, ...
Our contributions:

The rest of the paper is organized as follows. We briefly survey previous work on Gaming on Demand (GoD), parallel rendering and video encoding for the multiscreen services in Section 2. Section 3 describes the proposed the system architecture and the core systems in our system. We explain implementation details of our multi-view rendering algorithm and describe the performance result in Section **??**. In Section **??**, we compare our system and algorithm with prior GPU-based algorithms and highlight some of the benefits. Finally, we discuss future work and conclude in Section 7.

2. RELATED WORK

In this section, we give a brief overview of related work on the middleware for mobile computing and multiscreen services. We also highlight many technical characteristics of multiscreen user experience technology.

Many researchers agree in handling heterogeneity by employing middleware solutions. The key idea behind the middleware approaches is to position the middleware layer between the application layer at the top and the heterogeneous environments such as hardware and operating systems at the bottom.

3. SYSTEM ARCHITECTURE

In this section, we describe the proposed system architecture for the multiscreen-based collaboration services. Our architecture consists of two major systems such as the proposed *middleware* for smart devices and *N-screen service cloud* as shown in Fig. 1. Also, our middleware decomposes into 2 layers; the *N-screen application library* and *collaboration agent*.

3.1 System Overview

The Distributed Service Platform (DSP) is responsible for launching the game processes on the game execution nodes or rendering job on the Distributed Rendering System (DRS) after client-side invocation, monitoring its performance, allocating computing resources and managing user information. And, the DSP is responsible for processing user's game input via UDP from the client-side devices. In client-side, the user's game input is captured and transmitted via UDP by the user input capturing and transmission software on the client devices. Also, the DSP performs execution management of multiple games. In order to perform streaming the game A/V streams to the clients, the DSP requests capturing rendered frame buffer for video encoding and streaming to the Encoding/QoS/Streaming System (EQS).

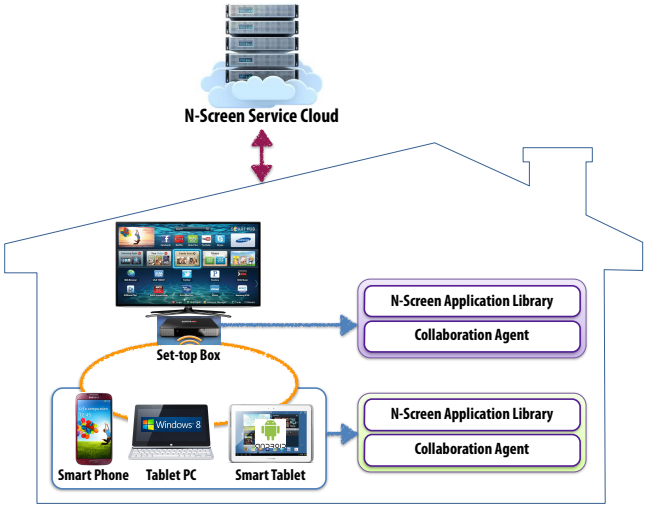

Figure 1: Our system architecture

3.2 Definitions

N-Screen Devices
Logical Application
Physical Application
Collaboration Session

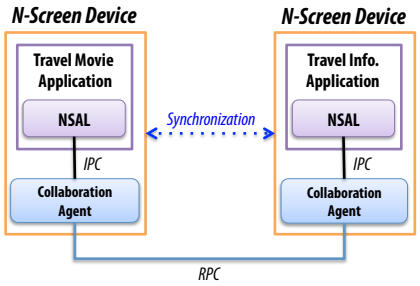

Figure 2: The CA block architecture

Representation
Inter-application relation schema

3.3 N-Screen Application Library

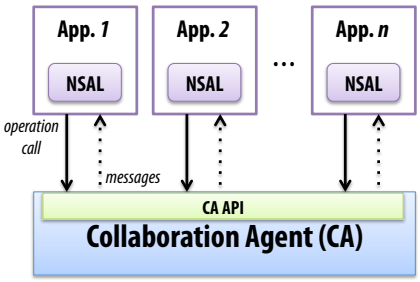

Figure 3: An example workflow for collaboration session construction

3.4 Collaboration Agent

Fundamental Operations:

- Service discovery
- Remote invocation
- Collaboration session join
- Collaboration session invitation
- Pull migration
- Push migration
- Synchronization

4. COLLABORATION SESSIONS

The collaboration sessions are roughly analogous to social organizations. The key approach to collaborating among the NSAL-based applications to organizing several interoperable applications into a group; we call this group a *collaboration session*. The purpose of introducing collaboration sessions is to allow n-screen applications to deal with collections of different smart apps as a single abstraction.

4.1 Representation

4.2 Persistency

4.3 Recovery with Device Substitution

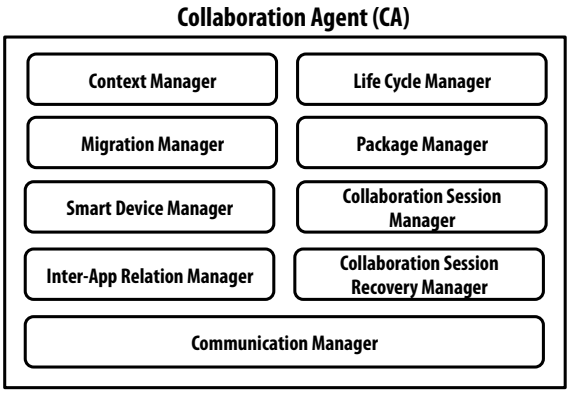

Figure 4: The CA block architecture

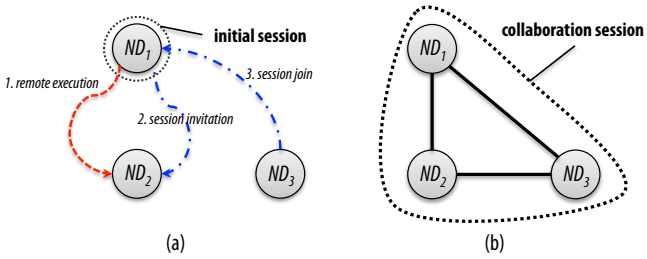

Figure 5: An example workflow for collaboration session construction

5. IMPLEMENTATION DETAILS

6. EXPERIMENTAL RESULTS

Analysis: Our rendering system provides good performance scaling of multi-core CPUs for multi-view rendering. And the multi-view rendering algorithm maps well to the current GPUs and we have evaluated its performance on two different GPUs with different rendering resolution. Furthermore, it is relatively simple to combine the video encoding methods and optimizations in the streaming-based gaming service framework. This makes it possible to develop a more flexible GPU-based framework for the video encoding methods like H.264/AVC or ORBX which is GPU-based encoding schemes.

Limitations: Our approach has some limitations. First, we support the multi-view rendering for one multi-user game, since it is

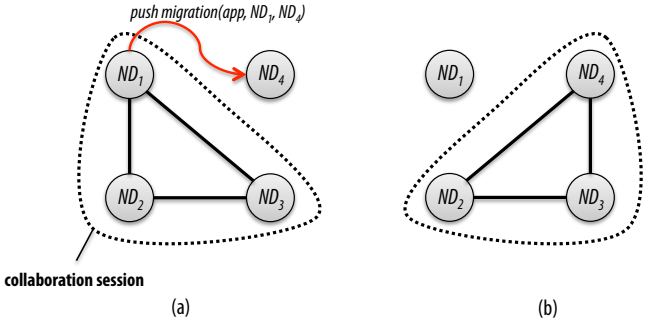

Figure 6: Collaboration session maintenance

Figure 7: Collaboration with multiple smart devices

difficult to share the rendering resources in a GPU among different games. We believe that this can be resolved by using multi-GPUs. Secondly, our system performs directly rendering to the framebuffers on the server-side machines. However, in terms of efficient services in the cloud-based gaming, we should exploit the *off-screen rendering* approaches and *GPU virtualization* techniques.

7. CONCLUSIONS

In this paper, we have presented a system architecture for the cloud-based gaming service and multi-view rendering. Our rendering system greatly improves utilization of hardware resources present in the system, allowing to utilize both multi-core CPUs and a GPU simultaneously.

We found that the proposed system provide the multi-view rendering for different focal positions for each viewpoint with high visual quality. Moreover, our approach is flexible and maps well to current GPUs in terms of shared resources such as textures and shaders for rendering. In addition, we demonstrate that the proposed rendering system could prove to be scalable in terms of parallel rendering. So, we believe that our rendering system will provide high-quality with good performance for the cloud-based gaming services.

There are many avenues for future work. It is possible to use new capabilities and optimizations to improve the performance of the video encoding especially H.264/AVC through the GPU-based implementation. Furthermore, we would like to develop algorithms for integrating the multi-view rendering with the video encoding in a GPU.

8. ACKNOWLEDGMENTS

This work was supported by ETRI R&D program ("*Development of Big Data Platform for Dual Mode Batch Query Analysis*, 14ZS1400") funded by the government of South Korea.

9. REFERENCES