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

Mobile middleware, Collaboration, Persistency

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. In case of the first-person shooting (FPS) game, the threshold of latency is below 100 msec 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 [**?**]. Also, the dedicated hardware and software for video encoding and streaming can allow users to share common resources.

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 6.

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. More specifically, our middleware decomposes into 2 layers; the n-screen application library (NSAL) and the collaboration agent (CA).

3.1 System Overview

Basically, our system architecture divided into two major systems as shown in Figure 2. First, our proposed architecture based on the *Smart Home* concept is targeted at the smart home environment in terms of *communication* and *socialization*.

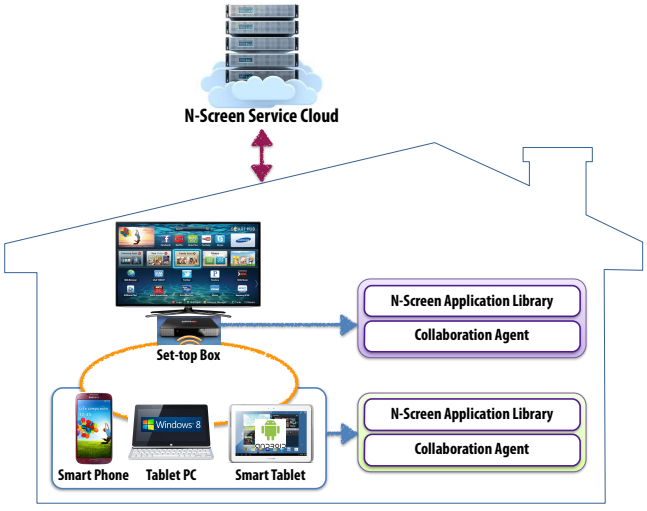

Figure 1: Our system architecture

In our work, the smart home consists of various wireless hosts such as smartphone, smart set-top, smart tablet, or laptop which connects each other through a wireless communication link in the same wireless network environment. Each wireless host has the *n-screen application library* and *collaboration agent*, which provide abstractions that reduce development effort and support interoperability between applications. These mobile devices communicate directly with each other in a *peer-to-peer* fashion with no centralized control. Hence, the *synchronization* for collaboration services is achieved through direct communication between applications.

Second, the n-screen service cloud is responsible for providing the collaboration applications similar to the apps in the *App Store*

and managing the application contexts for users. The primary benefit from the N-screen service cloud is *scalability*, *persistency* and *mobility* for the collaboration service. To represent the context of the application, we use the *key-value pair* which is the most simple context model. Context information of the applications and collaboration sessions are stored in the context repository at the service cloud.

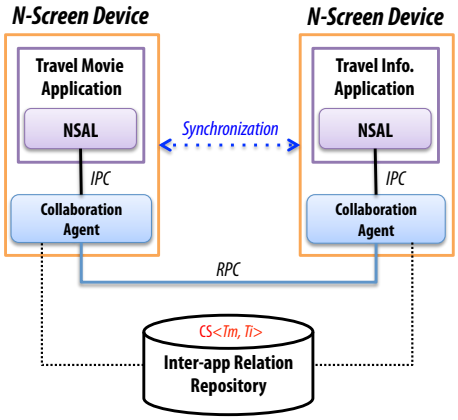

Figure 2: The CA block architecture

Here, we define major terminologies for our system architecture. The n-screen device ND is a wireless host which includes the NSAL and the collaboration agent as shown in Fig. 2. Every ND in the same network periodically sends UDP-based broadcast message for advertisement of their aliveness.

Applications: There are two kinds of application; *physical* and *logical* applications (or *app*). First, the *physical application*, A_p is an application running on each device in the same network. This lifecycle of A_p is the same as the android application and activity lifecycle. On the other hand, the *logical application*, A_l means an physical device-independent application at runtime regardless of physical device changing through application migration. Managing of logical application lifecycle plays an important role in our system. Even if a user migrates a application from one device to other device, the system should provide the the application service seamlessly and consistently. So, our system manages the logical application lifecycle to maintain the running states of application with contexts.

Collaboration sessions: Analogy to a social community, the *collaboration session*, $\mathcal{CS}$ is defined as a logical space that can be synchronized the associated information through the more than one physical applications at runtime. In order to represent a collaboration session, we exploit a graph-based representation $G(V, E)$, where V means a logical application set and E denotes a communication link set between two applications. An edge $e \in E$ is undirected and joins two vertices $v, u \in V$, denoted by (u, v) or (v, u). For instance, when each application like travel movie app or travel information app in Fig. 2 starts, the CA assigns a unique collaboration session ID and logical app ID for each application. If primitive collaboration operations such as session join and leave is processed, the CA will update the $\mathcal{CS}$. Fig. 2 shows the $\mathcal{CS}$ instance which consists of travel move app and travel information app. This $\mathcal{CS}$ will destroyed when all logical app in the same collaboration session is stopped at runtime. However, a user can save a $\mathcal{CS}$ as a persistent object to the n-screen service cloud during the runtime.

Inter-app relationship: Decoupling the collaboration information from the multiscreen applications requires representing the inter-application (or *inter-app*) relationship to support the scalable collaboration service. The CA includes a manager for describing the inter-app relation among the associated apps called *inter-app relation manager*. This manager uses an XML-based language to encode the necessary information for discovering, executing and collaborating among the associated apps. This inter-app relation information is stored on the n-screen service cloud. The inter-app relation manager in the CA periodically updates the information on logical storage through the RESTful API. A major benefit of the inter-app relation manager is providing the scalability in terms of the collaboration.

3.2 N-Screen Application Library

The n-screen application library (NSAL) is the library which provides common APIs for developing multiscreen-based collaboration application. The developers can implement an application through this interface of the NSAL. Each ND can have one CA and multiple NSAL-based applications. The NSAL provides the following functions:

- *Lifecycle event handling:* In order to support migration functions, the NSAL manages the lifecycle of the NSAL-derived objects which are inherited from the NSALActivity and the NSALApplication objects.
- *Describing the inter-app relation:* The developers can describe the inter-app relationship information for connecting the relations among the associated apps. This inter-app relation is represented by the XML-based schema. If the developers want to add an additional apps for extending the collaboration, they can insert information of the apps into the inter-app relation XML file.
- *Proxy for the CA interface:* The collaboration services which are based on the NSAL APIs can run by using the primitive operations in the CA. This operations can be obtained through the CA interface.

Since the collaboration management among the apps is handled by the CA, mobile devices are provided transparent access to a set of primitive services from the CA, thus successfully reducing the management complexity from them.

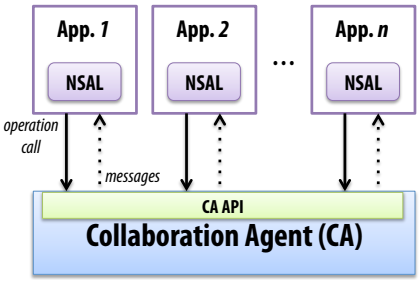

Figure 3: An example workflow for collaboration session construction

3.3 Collaboration Agent

The collaboration agent (CA) is a software component that acts for users or a NSAL-based apps in n-screen service environments.

Each n-screen device includes a CA as a singleton process. The CA consists of nine managers which provides functions such as device discovery, lifecycle management for n-screen apps, collaboration session management, messaging, and so on as shown in Figure 4.

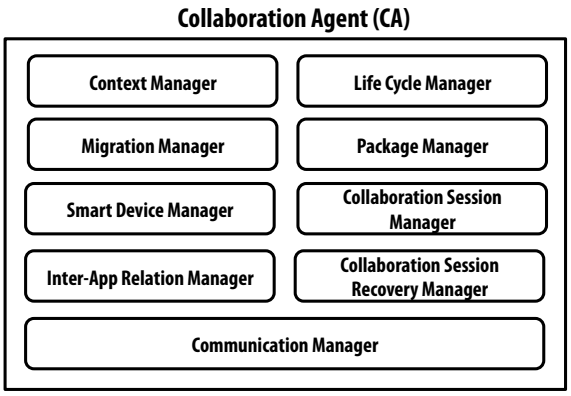

Figure 4: The CA block architecture

Fundamental operations: The CA provides the following key operations for multiscreen-based collaboration services.

- *Device discovery:* From the client point of view in our system environments, device discovery allows to discover dynamically n-screen devices present in the same network. The basic interactions among n-screen devices are *service advertisement* and *service discovery*. First, service advertisement allows n-screen devices to periodically announce their presence via the UDP-based broadcast after they enter the network. And then the CA provides the service discovery function via multicast messages in order to discover the n-screen devices in the network.

- *Remote execution:* User can execute the certain remote app at remote devices in the same network using a source device. For example, if a user want to execute the travel video app in remote set-top, a user can execute the remote travel video app via remote invocation using user's tablet. This function is useful to control diverse devices effectively.

- *Session join/invitation:* In order to make the collaboration among the associated apps, user can construct the collaboration session similar to social community. One of the associated apps in the collaboration session is allowed to join or leave the session. Moreover, user can invite the apps which are not in the collaboration session using the invitation function in order to collaborate and synchronize.

- *Application migration:* The CA provides the function which migrates the running apps from arbitrary device to other device at runtime. In our work, we exploit the migration function based on the strong mobility. Our system supports two types of app migrations; *push migration* and *pull migration*. Push migration is defined as source-initiated migration. In contrast, pull migration is destination-initiated migration.

- *Synchronization:* Events and messages exchanged dynamically among all apps in the same collaboration session. We use the TCP-based multicast messages for synchronization.

3.4 Collaboration Sessions

The collaboration sessions are roughly analogous to social organizations. The key approach to collaborating among the NSAL-based apps to organize 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.

Collaboration session contruction:

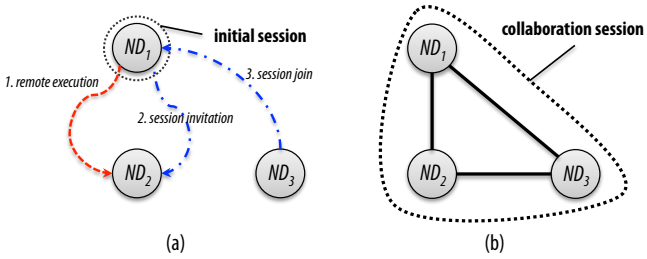

Figure 5: An example workflow for collaboration session construction

Persistent collaboration session:

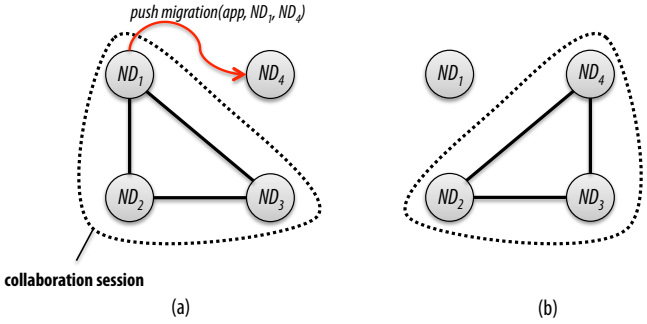

Figure 6: Collaboration session maintenance

Dynamic recovery with device substitution:

4. IMPLEMENTATION DETAILS

5. EXPERIMENTAL RESULTS

Figure 7: Collaboration with multiple smart devices

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 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.

6. CONCLUSIONS

Mobile technologies can potentially enhance social interactions and user's experiences, extend both social and informational resources available in context, and greatly alter the nature and quality of our interactions. In this paper, we have presented a system architecture for the cloud-based gaming service and multi-view rendering. Also, we have described a new technique for parallelized and distributed 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. without a CPU-GPU I/O for the video encoding.

7. 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.

8. REFERENCES