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

This paper focuses on the challenge that how users can receive the seamless collaboration services regardless of the changes of physical device configurations in the multiscreen environment. In order to solve this problem, we propose a novel system architecture which supports primitive operations, such as remote invocation, session join/invitation, push/pull migration and synchronization for collaboration among associated applications.

We introduce a mobile middleware which provides a collaboration service among associated apps in a symmetric fashion. The major advantages of our system are that the system can provides communication transparency, seamless collaboration services and scalability. The experimental results demonstrate that our system can be successfully applied to the collaboration services among multiple apps in the small network.

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 (or *apps*) running on various smart devices, such as smartphones and tablets. The main reason for this has been the realization that many diverse apps need a dedicated middleware for managing apps and collaborating among multiple associated apps [2, 10, 9, 3, 6, 8, 11, 5, 1, 4, 7]. In this paper, we propose a novel mobile middleware to support the seamless collaboration services among heterogenous multiple apps in diverse smart devices.

MidDep 2014, December 8-12, Bordeaux, France

To provide convincing the collaboration services in multiscreen environments, we describe the key requirements as follows:

- *Service Discovery*: Service discovery is one of the most important functions in order to collaborate among apps through remote invocation and session join/invitation in mobile computing environments. Since the wireless hosts in the wireless network are highly dynamic, service hosts should periodically announce their presence in the network.
- *Collaboration*: The system must support collaboration services among apps in smart devices. To satisfy this requirement, the system allows apps to join or leave a *collaboration session* which is a logical space being synchronized via associated information at runtime. However, collaboration session management in a peer-to-peer environment is quite challenging since its dynamic property.
- *Mobility*: The system must support app migration function, which migrates certain running apps from arbitrary device to other device at runtime.
- *Scalability*: The system must be able to scale with the growth in the number of apps for collaboration services. For example, it must support the dynamic session joins for new apps during a certain collaboration service is provided. So, it is necessary to develop the collaboration session management techniques to provide the scalability.

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

2. 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).

2.1 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*.

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

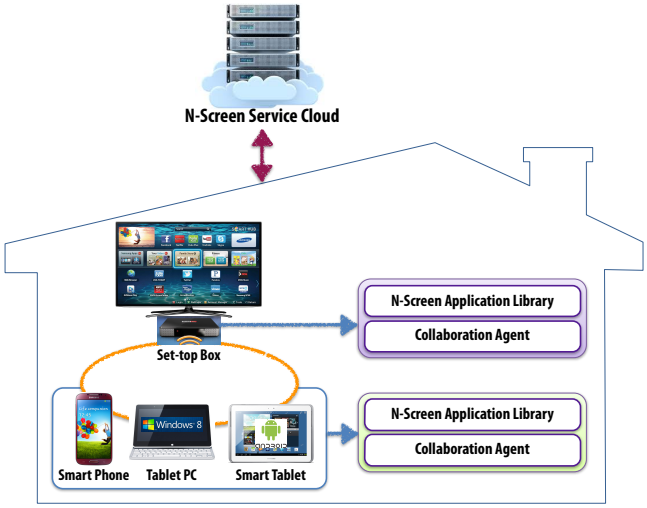

Figure 1: Our system architecture

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.

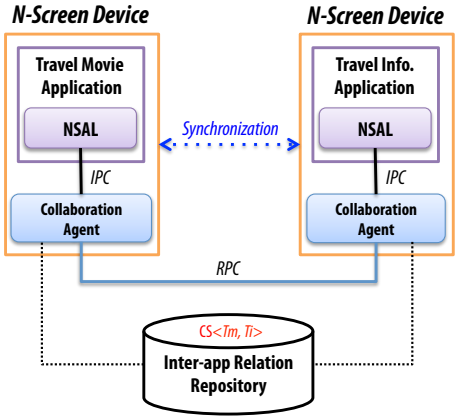

Figure 2: The CA block architecture

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.

N-Screen-based 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.

2.2 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 primi-

tive 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.

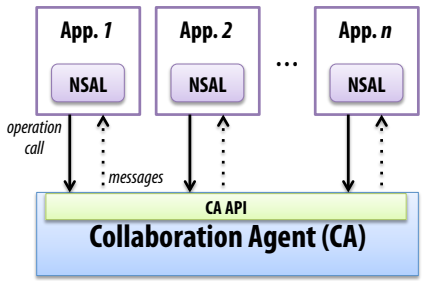

Figure 3: An example workflow for collaboration session construction

2.3 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.

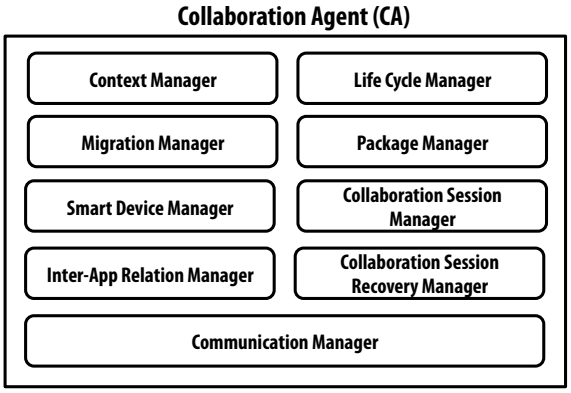

Figure 4: The CA block architecture

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 n-screen app residing in other n-screen 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 or smartphone. 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.

2.4 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 certain n-screen app to collaborate with collections of other smart apps as a single abstraction. Let ND_i be a i-th n-screen device, which has one or more n-screen apps and a CA.

Collaboration session contruction:

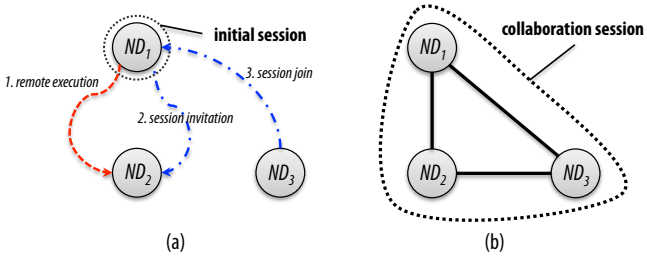

Figure 5: An example workflow for collaboration session construction

Persistent collaboration session:
Dynamic recovery with device substitution:

3. IMPLEMENTATION DETAILS

4. 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

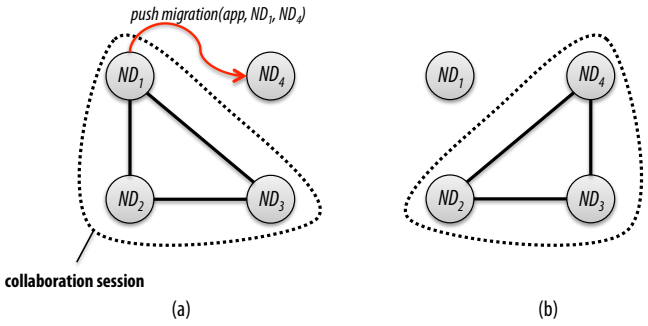

Figure 6: Collaboration session maintenance

Figure 7: Collaboration with multiple smart devices

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.

5. 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.

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.

6. CONCLUSIONS

We have presented a multiscreen service platform, a prototype system designs to provide collaboration services among associated apps in a smart home. The proposed collaboration middleware 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 interactions. Also, we have described a new technique for supporting a collaboration session persistency. Our middleware greatly improves the service discovery and binding performance, allowing to utilize UDP-based broadcasting and TCP-based messaging.

We found that the proposed system provide the scalability of the collaboration services by using the inter-app relationship representations. Moreover, our approach is flexible and maps well to various collaboration services in terms of extension of primitive operations, such as remote execution, collaboration session join/invitation and app migration. In addition, we demonstrate that the proposed system could prove to be flexible in terms of the interoperability among heterogenous apps. So, we believe that our middleware will provide the service scalability with good performance for the multiscreen-based collaboration services.

There are many avenues for future work. It is possible to use new capabilities and optimizations to improve the performance of low-level messaging technology for high-quality multimedia transmission. Furthermore, we would like to develop algorithms for dynamic device substitution when restoring a collaboration session.

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

[1] Samsung multiscreen sdk, http://multiscreen.samsung.com/, 2014.

[2] E. Aarts. Ambient intelligence: A multimedia perspective. *IEEE MultiMedia*, 11(1):12–19, Jan. 2004.

[3] E. Anstead, S. Benford, and R. J. Houghton. Many-screen viewing: Evaluating an olympics companion application. In *Proceedings of the 2014 ACM International Conference on Interactive Experiences for TV and Online Video*, TVX '14, pages 103–110, New York, NY, USA, 2014. ACM.

[4] L. Barkhuus and B. Brown. Unpacking the television: User practices around a changing technology. *ACM Trans. Comput.-Hum. Interact.*, 16(3):15:1–15:22, Sept. 2009.

[5] V. Fuentes, N. S. Pi, J. Carbó, and J. M. Molina. Reputation in user profiling for a context-aware multiagent system. In *EUMAS*, 2006.

[6] S. Longo, E. Kovacs, J. Franke, and M. Martin. Enriching shopping experiences with pervasive displays and smart things. In *Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication*, UbiComp '13 Adjunct, pages 991–998, New York, NY, USA, 2013. ACM.

[7] D. Ma, M. Liu, Y. Zhao, and C. Hu. Sscm: Middleware for structure-based service collaboration. In *Proceedings of the 2008 ACM Symposium on Applied Computing*, SAC '08, pages 2224–2225, New York, NY, USA, 2008. ACM.

[8] V. G. Motti and D. Raggett. Quill: A collaborative design assistant for cross platform web application user interfaces. In *Proceedings of the 22nd International Conference on World Wide Web Companion*, WWW '13 Companion, pages 3–6, Republic and Canton of Geneva, Switzerland, 2013. International World Wide Web Conferences Steering Committee.

[9] A. Nandakumar and J. Murray. Companion apps for long arc tv series: Supporting new viewers in complex storyworlds with tightly synchronized context-sensitive annotations. In *Proceedings of the 2014 ACM International Conference on*

Interactive Experiences for TV and Online Video, TVX '14, pages 3–10, New York, NY, USA, 2014. ACM.

[10] F. Sadri. Ambient intelligence: A survey. *ACM Comput. Surv.*, 43(4):36:1–36:66, Oct. 2011.

[11] R. Schmohl and U. Baumgarten. Heterogeneity in mobile computing environmens. In *ICWN*, pages 461–467, 2008.