A Collaboration Middleware for Peer-to-Peer Architecture

Sung-Soo Kim
Electronics and Telecommunications Research Insititute
Daejeon, South Korea
sungsoo@etri.re.kr

Jongho Won
Electronics and Telecommunications Research Insititute
Daejeon, South Korea
jhwon@etri.re.kr

ABSTRACT
We introduce a novel mobile middleware which provides a collaboration service among associated apps in a symmetric fashion. 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.

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
C.2.4 [**Computer-Communication Networks**]: Distributed Systems–Distributed Applications; D.2.7 [**Software Engineering**]: [Distribution, Maintenance, and Enhancement]

General Terms
Middleware, Mobile Computing

Keywords
Mobile Middleware, Collaboration Session

1. INTRODUCTION
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 [7]. However, previous methods related to multiscreen IPTV focused on transcoding the multimedia for adaptive content delivery. 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 [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12].

A second screen is a second smart device used by television viewers to connect to a program they're watching. A second screen is often a smartphone or tablet, where a special complementary app may allow the viewer to interact with a television program in a different way – the tablet or smartphone becomes a TV companion device. Prior research has focused on achieving

In this paper, we propose a novel mobile middleware to support the seamless collaboration services among heterogenous multiple apps in diverse smart devices. To provide convincing the collaboration services in multiscreen environments, we describe the key requirements as follows:

- *Service Discovery*: Since the wireless hosts in the wireless network are highly dynamic, service hosts should periodically announce their presence in the network. 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.
- *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 service scalability.

Our contributions: We present a novel system architecture for the multiscreen-based collaboration services, which utilizes the collaboration middleware for multiple associated apps in the same wireless network and the remote service cloud. The contributions of our work can be summarized as follows.

- *Collaboration middleware*: Our proposed middleware, which is based on peer-to-peer communication model, provides common APIs for diverse collaboration-based applications. Our

MidDep 2014, December 8-12, Bordeaux, France

middleware APIs include the primitive functions, such as, remote execution, session join/invitation and app migration.

- *Service scalability*: We introduce a schema for representing inter-app relationship in collaboration sessions. This model can help design collaboration services and add additional apps for service extension.

- *Lifecyle managment*: We propose a method for logical app lifecycle management to support seamless collaboration session.

The rest of the paper is organized as follows. Section 2 describes the proposed the system architecture and the core components in our system. We briefly survey previous work on multiscreen middleware, multiscreen IPTV, and second screen in Section 4. In Section 3, we presents the experimental results of our work in terms of service discovery. Finally, we discuss future work and conclude in Section 5.

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*.

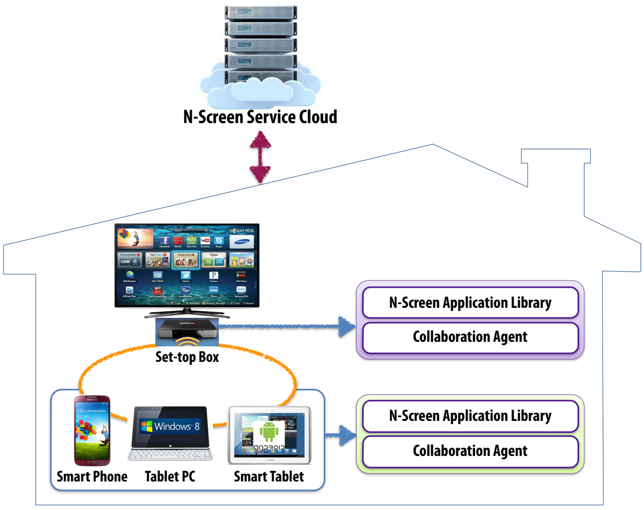

Figure 1: Our system architecture

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.

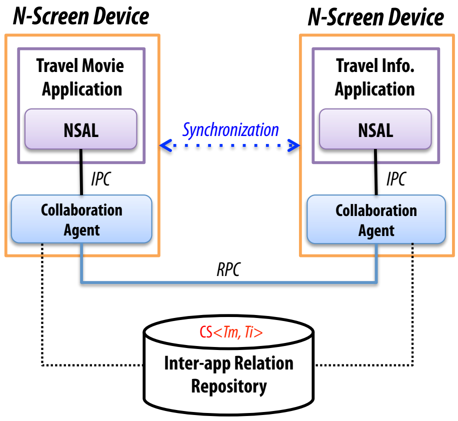

Figure 2: An example of n-screen collaboration service

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.

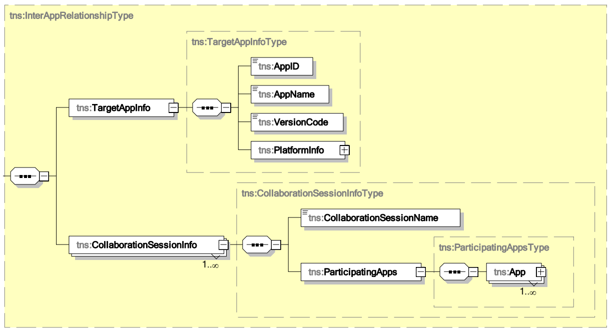

Figure 3: The schema diagram for inter-app relationship

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. Figure 3 shows the schema diagram for representing an inter-app relationship. We can describe the information of n-screen collaboration apps (TargetAppInfo) and their collaboration sessions (CollaborationSessionInfo).

This inter-app relationship 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 *service 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 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, n-screen apps are provided transparent access to a set of primitive services from the CA, thus successfully reducing the management complexity from them.

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.

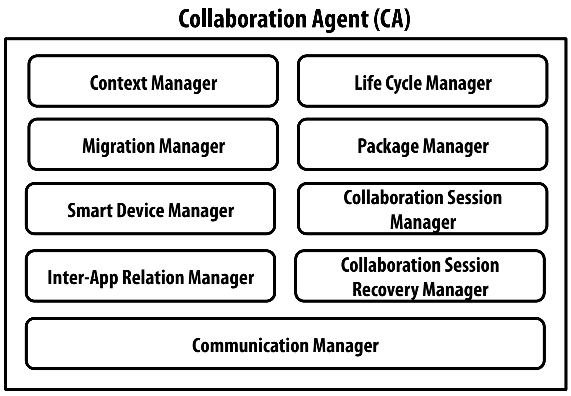

Figure 4: The CA block architecture

The context manager is responsible of managing contexts which are defined at n-screen applications. The migration manager handles the app migrations, such as, pull migration and push migration. The smart device manager deals with static (e.g. CPU and memory spec.) and dynamic profile (e.g. the amount of CPU and memory usages) of n-screen devices. The life cycle manager offers life cycle management for logical n-screen apps at runtime. And the collaboration session manager is responsible of maintaining the n-screen collaboration sessions. *Persistent collaboration session* is a reusable collaboration information at certain time, which is stored at the n-screen service cloud. Recovering a certain collaboration session is handled by the collaboration session recovery manager. These managers are singleton objects in a n-screen device.

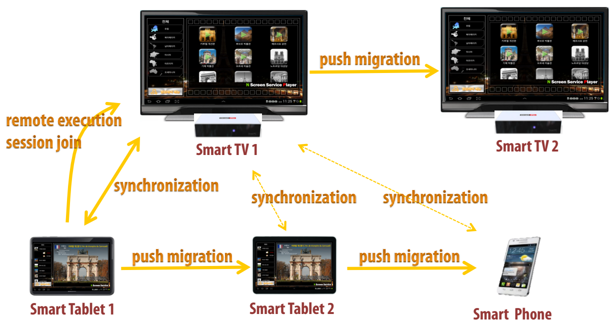

Figure 5: Fundamental operations for collaboration services

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 in the same network, which has one or more n-screen apps and a CA. Each ND has more than one n-screen apps, A_i, which are developed by using the NSAL APIs.

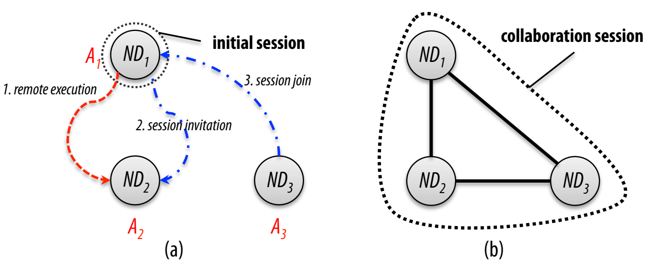

Figure 6: An example workflow for collaboration session construction

Collaboration session construction: Figure 6 shows an example workflow for collaboration session construction. Consider each n-screen device (ND_1, ND_2, ND_3) exists in the same network environment. First, user executes an n-screen app, A_1, in the ND_1. If the app is successfully started, NSAL creates an initial collaboration session as depicted in Figure 6(a). In order to perform the *remote execution*, A_1 requests the service discovery function to CA. This service discovery function can be called with service properties as parameters, such as, n-screen device or n-screen apps. Then A_1 invites the other app, A_2, in ND_2 through *session invitation* function. CA updates the collaboration session after session invitation is completed. If another app, A_3, in ND_3 want to join the collaboration session by using *session join* function, CA adds A_3's logical app information to the collaboration session. Finally, the collaboration session, which includes A_1, A_2, and A_3, is constructed as shown in Figure 6(b). The pseudo code for pipeline of this collaboration session construction is shown in **Algorithm 1**.

Algorithm 1 Collaboration session construction.

```
procedure ConstructSession
    CollaborationSession CS ← startApp(A1);
    A1.remoteExecution(ND2, A2);
    A1.sessionInvitation(A2);
    CS.add(A2);
    A3.sessionJoin(CS);
    CS.add(A3);
    return CS
end procedure
```

Seamless collaboration session: In our work, the multiscreen collaboration service consists of two apps; n-screen travel movie app and n-screen travel info app. The n-screen travel movie app provides guided movies related to attractions. The n-screen travel info provides various information of attractions, such as, basic information, POI data and maps. Figure 7 shows an example workflow for seamless collaboration session maintenance.

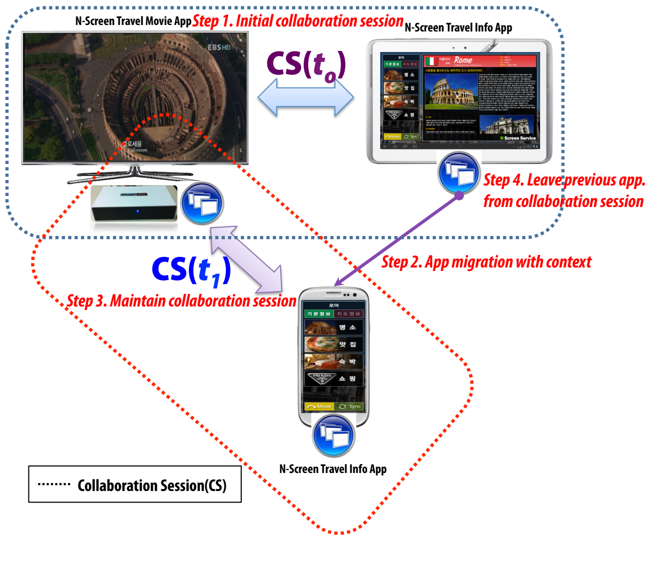

Figure 7: Seamless collaboration session

First, user constructs initial collaboration session, $\mathcal{CS}(t_0)$, using the collaboration operations in the CA. Then user requests push migration of n-screen travel info app from a smart tablet to a smartphone, the CA at the smart tablet sends the request message, which consists of app information with app context, to the CA in the smartphone. If this migration is done, the CA will update the collaboration session, $\mathcal{CS}(t_1)$. Finally, the physical n-screen travel app in the smart tablet leaves from collaboration session and stop the app.

3. EXPERIMENTAL RESULTS

In this section, we evaluate the performance of our system in terms of service discovery for the collaboration services.

We have implemented the CA and the NSAL in Java on Android. We conducted a series of experiments for our middleware performance on the multiscreen service platform. In order to test the performance of our system, we used two smart set-tops, which were connected to wireless network connection. Also, the three smart devices (one smartphone and two smart tablets) as mobile-clients were connected to the wireless network as shown in Figure 8.

Figure 8: Collaboration with multiple smart devices

In the experiments we tested the function of the n-screen resource discovery. We measured the processing time of service discovery function every minute during two hours to evaluate its performance. Figure 9 presents the processing times according to the n-screen resources. The results show that the proposed middleware provides good performance for the collaboration services.

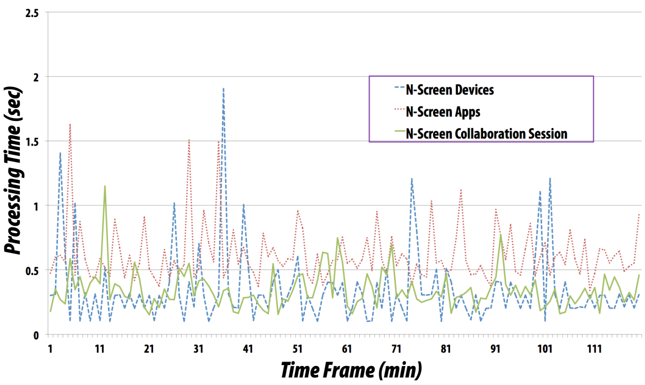

Figure 9: Performance result of n-screen resource discovery

Analysis: Our middleware provides good performance for service discovery of n-screen resources, such as, n-screen devices, n-screen apps, and n-screen collaboration sessions. And the proposed middleware maps well to the current smart devices and we have evaluated its performance on four different smart devices. Furthermore, it is relatively simple to deploy to smart devices on Windows platform as well as Android platform, since it is implemented in Java.

This makes it possible to develop a more flexible multiscreen application for a collaboration service.

Limitations: Our approach has some limitations. First, the collaboration agent (CA) uses JSON formats for exchanging messages between n-screen devices, thus it is difficult to exchange large messages between the CAs, such as, high-quality photos and videos. We believe that this can be resolved by n-screen application-side implementation. Secondly, the weakness of current system is a its lack of security. However, in terms of collaboration services in multiscreen environments, we should exploit the encryption methods for messages between n-screen apps and CAs, such as, data encryption standard (DES) and advanced encryption standard (AES), for improving the security.

4. RELATED WORK

In this section, we give a brief overview of related work on the middleware for multiscreen service, multiscreen IPTV and second screen.

Multiscreen Middleware: The potential of ambient intelligence (AmI) at home has been the subject of research for at least a decade in some major industries [2, 11]. 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 [12]. The multiscreen middleware in the AmI environment provides developers with a set of APIs and tools that optimize multiscreen experiences for various applications, such as, games, media sharing, social collaboration, and so on. In industry, Samsung has developed a multiscreen SDK for its smart TV products [1]. A multiscreen app based on this SDK provides separate views that are connected and running on different devices. All devices (and the TV) are connected, and can communicate with each other. However, this product supports *asymmetric collaboration* since this middleware requires a smart TV as a master in centralized client/server architecture to provide a collaboration service among associated apps.

Multiscreen IPTV: Many content delivery platforms are developed in order to provide adaptive content according to screen properties of smart devices [7]. These products focus on several considerations of multi-screen services in the context of over the top for multimedia content delivery. Prior researches investigated the changing television watching practices amongst early adopters of personal hard-disk video recorders and Internet downloading of video [6]. However, previous methods related to multiscreen IPTV focused on transcoding and streaming the multimedia for adaptive content delivery.

Second Screen: A cloud-based, multiscreen, social TV system can enrich content consumption and the TV viewing experience by incorporating and displaying geolocation-aware social data for users on a second screen. According to their survey, email and social media usage accounted for 20% of second screen interaction, and 25% of all second screen interaction focused on communication or information retrieval specifically about the show. The trend of users integrating second screen behaviours in their viewing habits, and practitioners interest in designing systems to support them has evolved a strong research agenda.

5. 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 system supports symmetric collaboration model as well as hierarchical collaboration in peer-to-peer network environment. And it provides seamless service using logical communication among apps and collaboration session management. 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.

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.

6. 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] A. Lucent. Shaing the future of multiscreen video. *Alcatel Lucent Strategic White Paper*, 2011.

[8] 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.

[9] 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.

[10] 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.

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

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