A Remote User Interface Framework for Collaborative Services Using Globally Internetworked Smart Appliances

Bongjin Oh*, Jongyoul Park *

*Bigdata S/W Platform, *ETRI, 218 Gajeong-ro, Yuseong-gu, Daejeon, 305-700, Korea*

bjoh@etri.re.kr, jongyoul@etri.re.kr

***Abstract*—This paper introduces a remote user interface framework which supports devices to share the UI of their applications with multiple smart devices. The smart devices are internetworked globally through RUI server. Besides that, a virtual IO function is provided to use mobile devices as remote controller. By thus, users can control home networked devices and applications by their smart devices with intuitive UI/UX. The proposed framework provides collaborative application model, APIs of sharing application view and virtual IO emulator.**

***Keyword*— RUI framework, home network, collaborative application, UI migration, Virtual IO**

I. INTRODUCTION

Smart devices, smart phones and smart pads, have become the most familiar appliances with users since those devices were introduced with multi-touch based intuitive user interface. TV is known for the friendliest consumer device to people. It also has evolved to smart device from a typical passive device.

The current TV provides users with various interactive contents augmented from linear services and downloaded from application servers. Moreover, it is not awkward for people to interoperate their smart devices with smart TV for collaboration services [2][5][7][8].

Recently, many researches have been introduced to interoperate the smart devices with other devices such as TV, information appliances and various sensors in home network area [1][3][4][6]. Many RUI (Remote User Interface) standards such as MIRACAST [11], DLNA-RVU [12] and Airplay [13] use streaming protocols to provide remote device control or collaborative services. They have some problems of using too a lot of bandwidth and supporting only sharing of main graphic user interface because they transmit video streams using the sequences of images captured from frame buffer.

Manuscript received Dec. 31, 2014. This work was supported by the IT R&D program of MKE/KEIT, [KI10039202, Development of SmartTV Device Collaborated Open Middleware and Remote User Interface Technology for N-Screen Service]. Bongjin Oh is with the Electronics and Telecommunications Research Institute, 218 Gajeong-ro, Yuseong-gu, Daejeon, 305-700, Korea (corresponding author to provide phone: +82-42-860-6384; fax: +82-42-860-5885; e-mail: bjoh@ etri.re.kr).

Jongyoul Park was with the Electronics and Telecommunications Research Institute, Daejeon, 305-700, Korea. (e-mail: jongyoul@etri.re.kr).

An HTML5 based collaborative application platform is provided by MOVL UI [14]. It is independent of device platform and based on a cloud server for collaboration services. But it is time consuming for users to connect client applications with host applications. Multiple applications should be installed on smart devices, and users should interconnect the devices by logging into allocated room with room number displayed on TV screen by host applications.

This paper proposes a RUI framework based on sharable GUI to support collaborative services among interconnected smart devices. Virtual IO emulator is also provided to control remote devices using virtualized device controllers.

The rest of this paper is organized as follows. Chapter II describes the overview of the proposed RUI architecture, and a reference implementation of the RUI framework with exemplary RUI services is shown in chapter III. Lastly, we conclude our research briefly in chapter IV.

II. THE PROPOSED RUI FRAMEWORK

A. *Network Configuration of the RUI Framework*

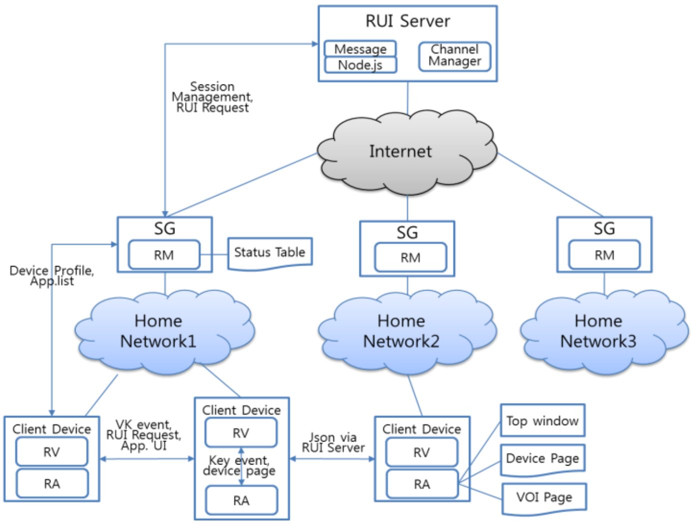


Fig. 1. The proposed RUI network configuration. There is a RUI server to interconnect all of home networked devices.

The RUI network is configured as Fig. 1. One SG (Service Gateway) and several fixed or mobile smart devices are internetworked in each home network. Smart devices control and share UIs with other devices using RUI framework. SGs are interconnected with each other through RUI server to support smart devices to control and share UIs with remote devices located at different home networks.

Service gateways manage devices and services of each home networks. Each SG also makes a channel in RUI server to allow remote devices to access local devices managed by them. The channel is set up with SG ID, login and password.

The smart devices can't send RUI messages directly to target devices because the most of them are networked with VPN (Virtual Private Network). Therefore, they require local SGs connected with them by same home network to forward their requests, formed as RUI messages, to remote devices. SGs also forward the requests to remote SGs connected with the target devices with the help of RUI server.

RUI server manages channels with ID, login and password as well as IP address for SGs, and forward the received RUI messages to SG designated with SG ID after verifying login and password included into RUI messages. For this, RUI server also support asynchronous HTTP based communication protocol for SGs to send and receive forwarded RUI messages asynchronously.

Below diagram describes the procedure of making a session and exchanging RUI messages among RUI devices through RUI server.

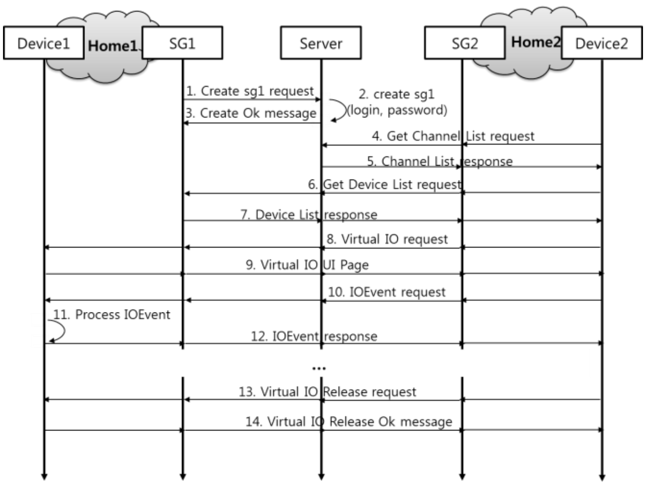


Fig. 2. The procedure of communication among RUI devices located at different home networks. RUI server manages channels of SGs to forward the input RUI messages to target devices.

B. Components of the RUI framework

Three kinds of RUI components are launched on the home networked devices such as SG and client devices. Their functions are described at Table I.

TABLE I
COMPONENTS OF THE PROPOSED RUI FRAMEWORK

Components	Function
RUI Manager	- manages the home networked devices and RUI services - provides local devices with method to communicate with devices outside home
RUI Agent	- manages the UIs of launched applications and virtual IO page - collaborates with other devices for sharing of UI and IO functions
RUI Viewer	- displays application UI and virtual IO UI - handles the user events

RM (RUI Manager) is the only component which runs on each service gateways and RA (RUI Agent) and RV (RUI Viewer) can run on every smart device.

The RM scans the RUI devices using the SSDP (Service Discovery Protocol) of UPnP. Whenever a RUI device turns on, The RA also find the RM using SSDP. If two devices are connected, then the RM collects device profile from the RUI device. The device profile includes device name, service list, address and device mode, and it is stored to RUI Status Table. The RM monitors the status of RUI devices. Client devices get the global information for RUI services such as RUI application list, RUI devices, from the RM. The status of client devices including current running RUI application and connection mode is stored to RUI Status Table together with device profile s. The RM also supports service session management among client devices. If a client device is gone, then the RM notifies it to another client device bound to disappeared client device.

The RV is implemented by extension of the WebKit to render HTML5 based UIs, and it handles user events invoked locally or remotely. The user events are transmitted to the RV's event queue by the RA whenever users input events with local input device or remote virtual input devices. The RV is launched automatically to render initial RUI Page by the RA.

The RA plays the most important role of RUI framework to share UIs for remote control of devices and applications among smart devices. The RA manages Top Window, Device Page, Virtual IO Page, RUI applications and local repository and so on.

The Top Window is displayed as an overlay window on the screen for interaction between users and RUI components. Users can request RUI operations of the RA by long touch on the top window for about 3 seconds. The RA shows the functions such as virtual IO on the Top Window, and then users select one of them to process.

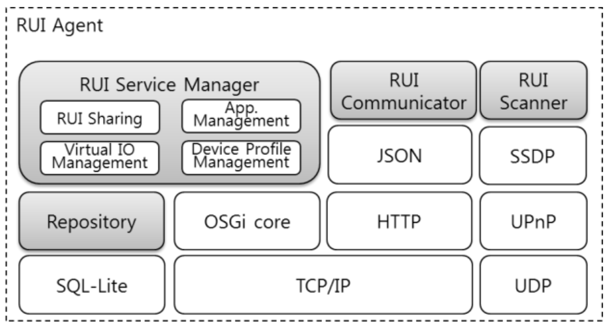


Fig. 3. The protocol stack of the RUI Agent.

The software elements of RUI Framework except the RV interoperate with each other using HTTP based messages. The RA provides the software elements with RUI Communicator APIs based on JSON message system. On the contrary, The RV is tightly coupled with the RA on every client devices, and they interoperate with each other using local procedure calls to invoke user's input events and to render application's or RUI initial UI page.

Virtual IO Page is also described as an HTML5 document including the key map of physical controllers such as remocon, mouse and keyboard provided locally. Virtual IO Page is transmitted to other Remote Agents for virtual IO mode. The Remote Agent shows the received Virtual IO Page to users by the RV to control remote devices using the similar user interface of physical controller.

The Device Page is changed according to current mode of client's device. When the RUI framework is launched, the page is set with the RUI Initial Page. If the device is bound to other device as virtual IO mode, then the page is set with the Virtual IO Page. Lastly, the page can be set with UIs of RUI applications launched locally or remotely. If the RUI applications are running on a remote client, then the UI to be set is moved to local device from the remote client.

The RA installs RUI applications to the repository of client devices, and manages their life-cycles. It also request remote RAs to launch RUI Applications installed on the remote devices. The UIs of RUI Applications are able to be migrated into other devices for remote control. The UIs and Virtual IO Pages are distributed to remote RUI Agents using OSGI's core APIs. In this paper, the UIs and Virtual IO Pages are handled as sub-apps included in a service bundle.

C. Collaboration Model of RUI Application

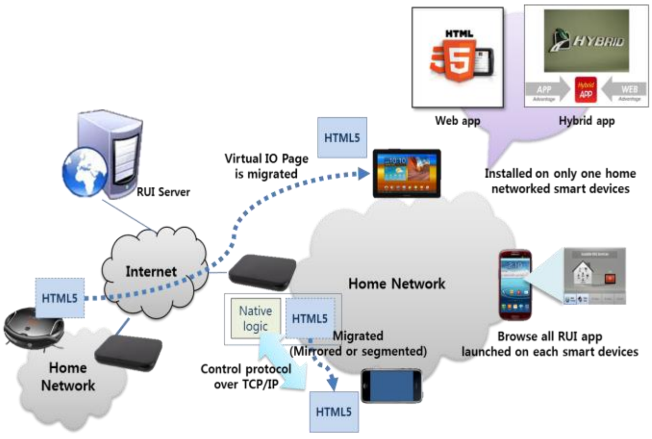


Fig. 4. The concept diagram of the proposed collaboration model. HTML5 based UI is composed of serveral parts to be shared by serveral devices.

RUI based services are installed on one of inter-connected smart devices by users. Users can browse the service list regardless of local service or remote service by RUI service browser. When users select a service, the selected service will be launched on the device which it is installed on. Users can control the remote service by the proposed RUI protocol based on migratable UI as Figure 5.

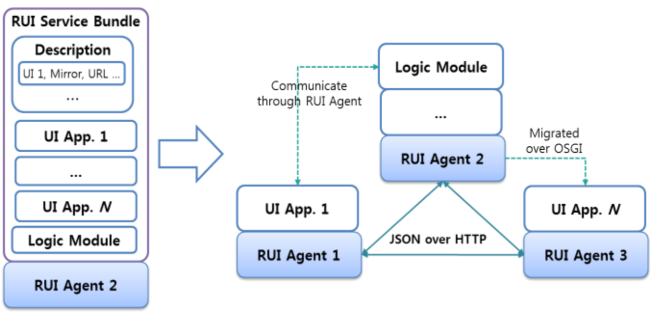


Fig. 5. The collabroation model of RUI applications. Migrated UI segments will communicate to the logic installed in local device through RA.

The Sub-description of each UI segments includes attributes such as segment ID, sharing mode, URL and input event model etc. Three kinds of the UI sharing modes are provided as follows.

1) Mirror

The UI of the original application which runs on the remote device is duplicated, and the UIs are transmitted to multiple devices. If an input event is invoked in the original UI, then all the duplicated UIs also receive the invoked event at the same time.

2) Migration

The original UI of local device is moved to selected devices. This mode is needed to display the local UI on the bigger screen. The local UI is automatically changed into virtual IO mode to control the migrated UI with local device.

3) Segmentation

The parts of Remote UIs are pulled and rendered on the display of local device. The UIs may be displayed on multiple devices according to the requests of several users at the same time.

As described in Fig. 2, Virtual IO page can be transmitted to globally networked devices for remote control through RUI server. The RUI messages of Virtual IO Page and invoked user events are routed to remote devices through local SG, RUI server and remote SG. Because of the problem of synchronization, the UI of application can't be shared between two devices located at different home until now.

The migrated UIs and logic communicated to each other with RUI messages which have the format as Fig. 6.

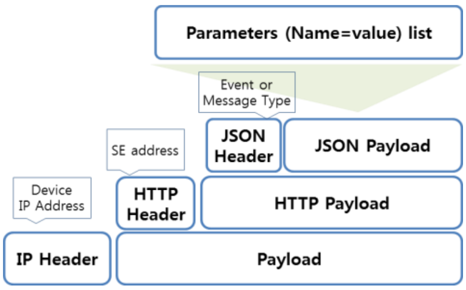


Fig. 6. The Structure of RUI Message. SE address contains ID to distinguish which target is local devices or remote devices and target software element.

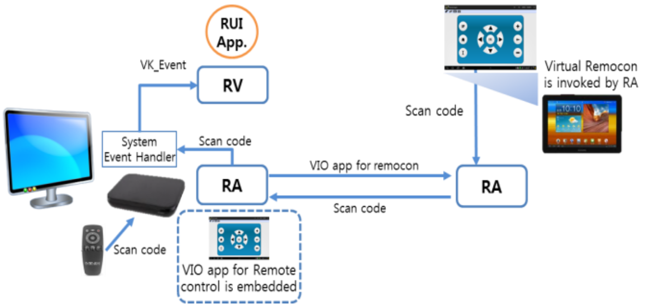
The address of RUI message consists of SG ID, device UUID, software element ID and segment ID to distinguish software elements. The kinds of software elements are classified into RUI applications (UI segments, logic apps) and RUI components (RM, RAs).

The IP address of RUI message is decided by the related information stored to the RUI Status Table such as the composition of device ID, application ID and UI segment ID (application ID is allocated per RUI service and included in service descriptions).

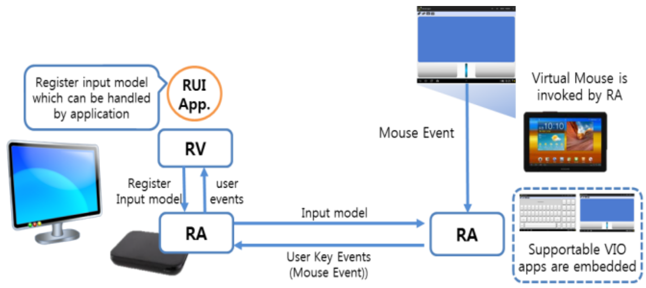
A RA figures out the SG ID is local or not before transmission of RUI messages. If SG ID is for local SG, then the RA transmits RUI messages to target elements directly. Otherwise, the messages are redirected to target elements of globally networked devices via RUI server. First, RA tosses RUI messages to local RM to forward those messages to target elements. Then, Local RM transmits RUI messages to RUI server to be verified and forwarded to target RM. lastly, the remote RM transmits the received messages to target device using IP address mapped to device ID contains in the messages.

D. Virtualized Input Devices

The proposed RUI framework provides two virtual IO modes for users to utilize smart devices as remote controllers as shown in Figure 7.



(a) Device based virtual IO mode



(b) Application based virtual IO mode

Fig. 7. Virtual IO action model. Virtaul IO Page is pulled to local device, so users can control the remote device as local device.

The first mode is the device virtual IO mode which user device plays the role as same as physical controller of other smart devices. Each RA manages Virtual IO Page which describes control UI of local device's physical controller such as remocon or control panels. The Virtual IO Page is described as HTML5 based application which can be rendered by the RVs of other devices. The Virtual IO Page is transmitted to other devices and launched by RVs of the devices, when users want to control remote devices by virtual IO mode.

User inputs are transmitted to remote RA as remocon control codes, and the codes will be consumed by system event handler. The system handler forwards codes to user event handler after converting them into key events. This procedure is also processed when user press a button of physical remocon. If any smart appliances, such as refrigerator, boiler and washing machine as well as TV, can be accessed from internet, then user can remotely control them using device based virtual IO mode at any home.

The second mode is the application virtual IO mode which user can select necessary type of virtual controller dynamically. RA manages embedded virtual IO emulators to be launched by only local RV, not by remote RVs. Virtual IO emulators are predefined in the repository of the RA according to the capabilities of local devices.

RUI applications should notify the type of input model for control their UI segments to RAs which launch the UI segments according to user's request, and RAs will provide users with selectable IO emulators among embedded emulators. If an event model contains keyboard and mouse, then the icon of keyboard and mouse is displayed in the virtual IO menu to be selected by users.

If a virtual mouse emulator is selected then user's inputs are transferred to remote RA through local RA as Mouse Events. Then the remote RA forwards the Mouse Events to RV, and RUI Apps (UI Segments) consume the Mouse Events generated by RV. This procedure is different with that of physical remocon.

III. Reference Implementation

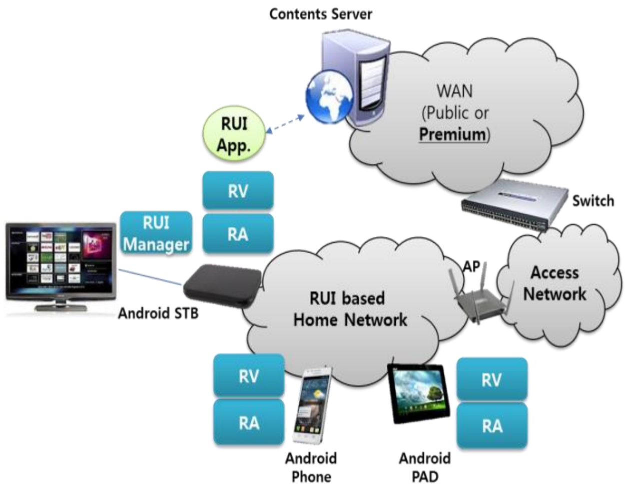


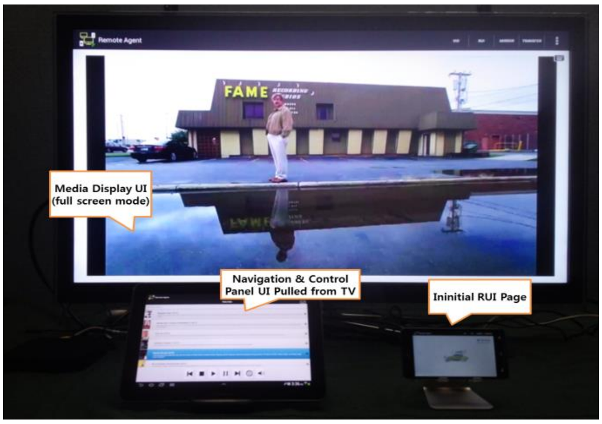
Fig. 8. Network configuration of a reference implementation. All devices are interconnected with one home network except contents servers.

A reference platform is implemented to show the functionalities of the proposed RUI framework together with an exemplary RUI application. The network configuration of the reference platform is shown as Figure 8 and Table II.

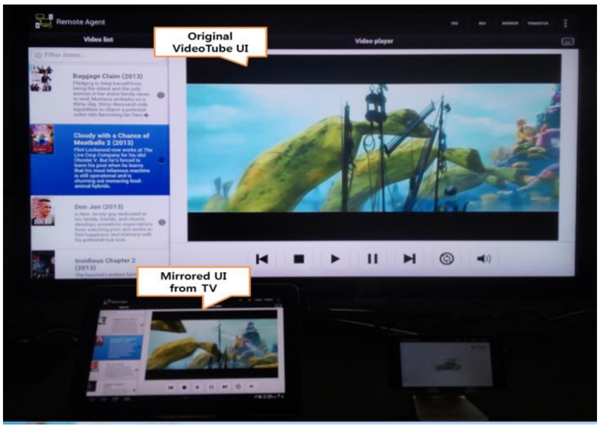
TABLE II
The Details of Device and Software Environment

Item	Details
Hardware	- Set-top box, smart phone, smart pad Android 2.x.x, dual core, RAM:2GB - Contents Server (PC), Windows 7, quad core: i-7, RAM: 4GB, HDD: 1T, 5400RPM
Networks	- AP (WiFi n/g, Ethernet 100Mbps) and switch (1Gbps)
Software	- VideoTube RUI application (VOD client) media player, contents guide, media control functionalities (UPnP AV renderer) - VideoTube server UPnP AV architecture Directory service, HTTP based streamer - HTML5 based web apps - handles the user events

Android based a set-top box; a smart phone and a smart pad are interconnected by an AP connected to a Giga-bit switch. A VOD server is connected directly to switch as an UPnP AV server. VOD client is installed the set-top box as an UPnP AV Renderer. The RM runs on the smart pad to manage the status of RUI framework. Some HTML5 games found on websites by the keyword of "HTML5 games" are installed on both of set-top box and smart pad. The smart phone is only used to control other devices as a virtual IO or remote UI sharing mode.



(a) Segmenting RUI mode



(b) Mirroring RUI mode

Fig. 9. A Exemplary RUI service (Segmenting & Mirroring).

VideoTube RUI service was implemented for remote UI sharing functionality among home-networked devices as Figure 9. VideoTube is composed of a contents server and a media player to provide users with a VOD-like service.

The UI of media player is able to be fragmented into Media Display UI, Control Panel UI and Contents Navigation UI. In the (a) of Figure 9, a user pulled the Contents Navigation UI together with Control Panel UI from TV. Only Media Play UI remains on the screen of TV with full screen mode automatically. There are 8 kinds of layout templates are provided for various status of RUI sharing for VideoTube.

The mirroring RUI mode is shown in the (b) of Figure 9. The UIs of VideoTube are duplicated to smart pad, and users can control the VideoTube by local smart pad or remote TV. The user input events are multicast to all of mirrored devices as well as TV which launches the VideoTube.

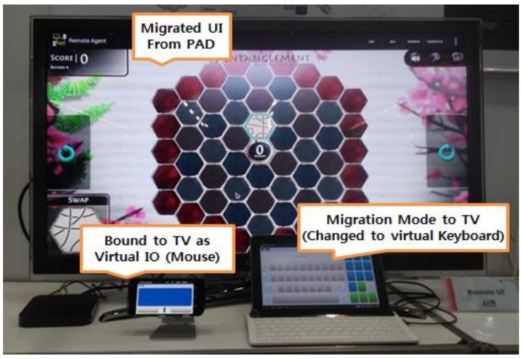


Fig. 10. An Exemplary Virtual IO service (Mouse & Keyboard)

Some HTML5 based web apps deployed on websites are used to verify our proposed virtual IO functionalities. Some of them are developed for PC, and the others are developed for mobile devices. Therefore, user's smart device should be virtualized into keyboard for PC version and virtualized into mouse for mobile device version according to user's selection. Fig. 10 shows an example of virtual IO service collaborated with TV and two smart devices.

An Entanglement game [15] is launched in PAD. When user requests the game migrate to TV, the UI of user's PAD is changed to the virtual Keyboard automatically. The phone is also bound to TV as a virtual mouse mode. Two users can play the game together simultaneously.

IV. Conclusion

The proposed RUI framework supports collaborative services using decomposable and sharable UIs among interconnected smart devices. The framework consumes less network bandwidths than the typical streaming based RUI protocols because the RUI framework uses HTML5 based UI and message driven interoperation between multiple devices, Moreover, users can use local devices as intuitive remote controllers of other devices using virtual IO emulators. Those functionalities can be provided to devices interconnected with different home networks through the RUI server using SG channels.

TABLE III
The Comparison of RUI Protocols

Function	Proposed	MIRACAST	Airplay	MOVL UI
Bandwidth	**Low**	High	High	Low
Mirroring	**OK**	OK	OK	NO
Collaboration	**OK**	NO	OK	OK
Virtual IO	**OK**	NO	NO	NO
Device Paring	**Easy**	Easy	Easy	Difficult
Platform Independent	**OK**	OK	NO	OK

Table Ⅲ shows that the proposed RUI framework is better than other RUI standards for various RUI functionalities.

References

[1] Bong-Jin Oh, Jong-Youl Park, "A Remote User Interface Framework for Collaborative Services Using Interconnected Smart Devices," *in Proc. ICACT2014*, Pyunchang, 2014, pp.630-634.

[2] Moon Soo Lee, Seung-Joon Kwon, "Lightweight Inter-device Interface Control System for the Machine-to-Machine (M2M) Interaction in the Internet of Things," *in Proc. ICCC2014*, Jeju Island, 2014, pp.481-482.

[3] Bumsuk Choi, Junghak Kim, Soonchoul Kim, Youngho Jeong, Jin Woo Hong, and Won Don Lee , "A Metadata Design for Augmented Broadcasting and Testbed System Implementation," *ETRI Journal*, vol.35, No.2, pp. 292–300, Apr. 2013.

[4] Kyoung Ill Kim, Su Young Bae, Dong Chun Lee, Chang Sik Cho, Hun Joo Lee, and Kyu Chul Lee, "Cloud-Based Gaming Platform Supporting Multiple Devices," *ETRI Journal*, vol.35, No.6, pp. 960-968, Dec. 2013.

[5] L. Bassbouss, M. Tritschler, S. Steglich, K. Tanaka, and Y. Miyazaki, "Towards a Multi-Screen Application Model for the Web," *in Proc. the 1st IEEE International Workshop on Consumer Devices and Systems (CDS 2013)*, Kyoto, 2013, pp.528-533.

[6] Steven Morris, Anthony Smith-Chaigneau, *Interactive TV Standards*. Elsevier Inc., 2005.

[7] Moon Soo Lee, Min Jung Kim, Sun Joong Kim, Cho Kee Seong, "Remote Collaboration Screen Control Using Mobile Multi-Touch Interface," *in Proc. ICTC2012*, Jeju Island, 2012, pp. 272-273.

[8] Yusoek Bae and Jongyoul Park, "A Seamless Remote User Interface System Supporting Multi-Screen Services in Smart Devices," *in Proc. ICCE2013*, Lasvegas, 2013, pp. 462–463.

[9] Bong-Jin Oh and Jong-Youl Park, "Design and Implementation of HTML5 based Collaborative N-Screen Contents Platform for Smart TV," in *Proc. ICCC2013*, Okinawa, 2013, pp.225–226.

[10] JeaWon Moon, Tae-Beom Lim, Kyung Won Kim, Seok Pil Lee, SeWoom Lee, "Advanced Responsive Web Framework based on MPEG-21," *in Proc. ICCE-Berlin2012*, Berlin, 2012, pp.192–194.

[11] Miracast. [Online]. Available: https://www.wi-fi.org/

[12] RVU Alliance. [Online]. Available: http://www.rvualliance.org/

[13] Apple AirPlay. [Online]. Available: http://www.apple.com/airplay/

[14] MOVL UI. [Online]. Available: http://connect.movl.com/

[15] Entanglement HTML5 based web app. [Online]. Available: http://entanglement.gopherwoodstudios.com

Bong-Jin Oh received B.S. and M.S. degrees in computer science from Pusan National University, Busan, Korea in 1993 and 1995 respectively, and the Ph.D. degree from Chungnam National University, Daejeon, Korea in February 2012. Since 1995, he has been with the Electronics and Telecommunications Research Institute (ETRI), where he develops home network middleware and data broadcasting middleware. His research interests are home network middleware, data broadcasting middleware, IPTV, pervasive computing, and big data analytics.

Jongyoul Park received the B.S. degree in computer engineering from Chungnam National University, Korea, in 1996, the M.S. and Ph.D. degrees in information and communication engineering from the Gwangju Institute of Science and Technology (GIST), Korea, in 1999 and 2004, respectively. From 2001 to 2002, he was a visiting researcher at the school of computing, University of Utah. Since 2004, he has been a Research Staff and Director of Analytics SW Research Section of Electronics and Telecommunications Research Institute (ETRI), Korea. His research interest includes IP broadcasting, software middleware, mobile code, distributed computing, big data and analytics platform.