A Remote User Interface Framework for Collaborative Services Using Interconnected Smart Devices

Bong-Jin Oh*, Jong-Youl Park*

* Social Computing Research Team, ETRI, 218 Gajeongno, 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 applications to share their view with remote smart devices. 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.**

***Keywords*— RUI framework; home network; collaborative application; UI migration, virtual IO**

I. INTRODUCTION

Smart devices are widely spread to users since the smart phone with intuitive UI/UX has been introduced in January 2007. 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 [1-3]. Moreover, it is not awkward for people to interoperate their smart devices with smart TV for collaboration services [4-6].

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. Virtual Desktop Interface based user interfaces have been used to control remote devices by local smart devices. Virtual Network Control, Remote FX and streaming protocol have been used to support those Virtual Desktop Interface services [4, 5]. Many Remote User Interface standards such as MIRACAST [7], DLNA-RVU [8] and Airplay [9] 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.

An HTML5 based collaborative application platform is provided by MOVL UI [10]. 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. RUI Framework Architecture

The proposed RUI framework consists of three RUI components shown as Figure 1. RUI Manager is the one component which runs on the server device. RUI Agent and RUI Viewer are the components which run on every client devices.

The RUI manager scans the RUI devices using the SSDP (Service Discovery Protocol) of UPnP. Whenever a RUI based smart device is turned on, the RUI Agent of the device finds the RUI Manager using SSDP too. If the two devices are connected, then the RUI Manager collects device profile of client device.

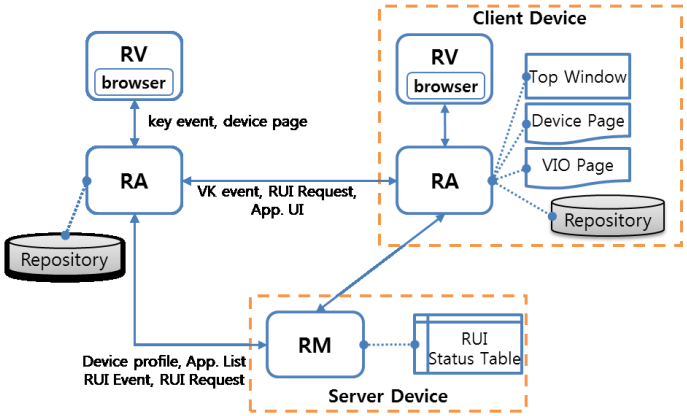


Figure 1. RUI framework architecture

The device profile includes device name, service list, address and device mode, and it is stored to RUI Status Table. The RUI Manager monitors the status of each client devices in real-time. Client devices get the global information about RUI services from the RUI Manager. The status of client devices including current running RUI application and connection mode is stored to RUI Status Table together with device profiles. The user can monitor the entire status of RUI devices using the management screen provided by the RUI Manager. The RUI Manager also supports service session management among client devices. If a client device is gone, then the RUI

Manager notifies it to another client device bound to disappeared client device using RUI messages.

The RUI Viewer 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 RUI Viewer's event queue by the RUI Agent whenever users input events with local input device or remote virtual input devices. The RUI Viewer is launched automatically to render initial RUI Page by the RUI Agent.

The RUI Agent plays the most important role of RUI framework to share UIs for remote control of devices and applications among smart devices. The RUI Agent manages Top Window, RUI Message layer, 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 RUI Agent by long touch on the top window for about 3 seconds. The RUI Agent shows the functions such as virtual IO on the Top Window, and then users select one of them to process.

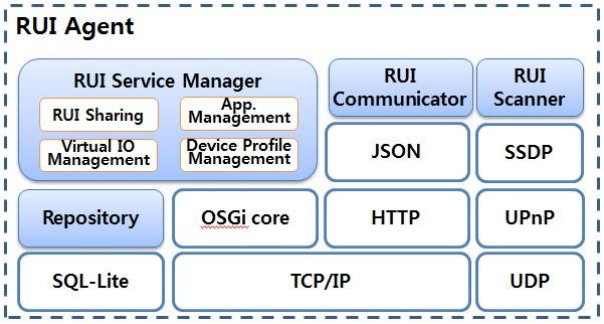


Figure 2. The protoco stack of the RUI Agent

The software elements of RUI Framework except the RUI Viewer interoperate with each other using HTTP based messages. The RUI Agent provides the software elements with RUI Communicator APIs based on JSON message system (Refer to Figure 2). On the contrary, The RUI Viewer is tightly coupled with the RUI Agent 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 RUI Viewer 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 RUI Agent installs RUI applications to the repository of client devices, and manages their life-cycles. It also request remote RUI Agents 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.

B. Collaboration Model of RUI Application

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 3.

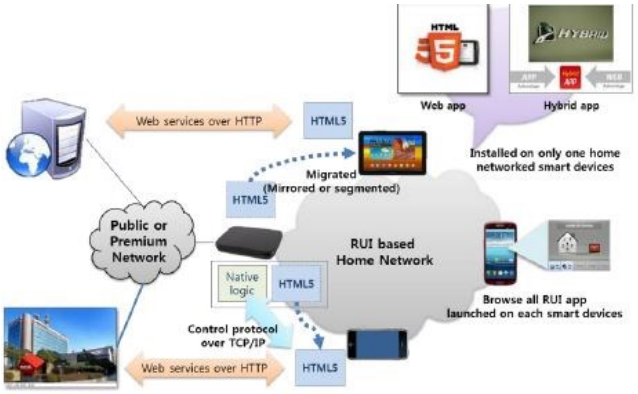


Figure 3. The concept diagram of the propsed sharable UI

In this paper, the RUI services are implemented as Web App or Hybrid App. Therefore, their UIs are described as HTML5 document to be rendered on various kinds of device platform by the RUI Viewer as Figure 4. RUI service bundle consists of a service description, several UI segments and a logic app, and the UI Segments can be distributed into smart devices according to user's requests.

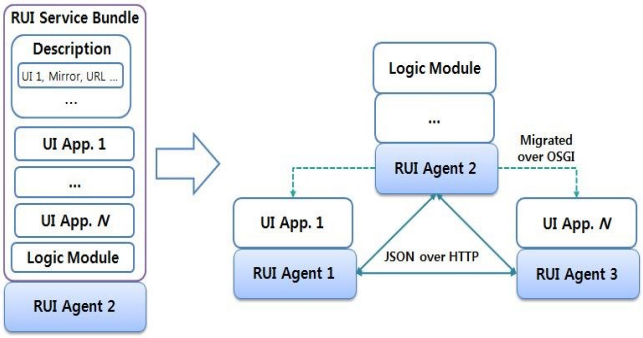


Figure 4. The collabroation model of RUI applications

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.

The shared UIs and logic communicated to each other with RUI messages which have the following format.

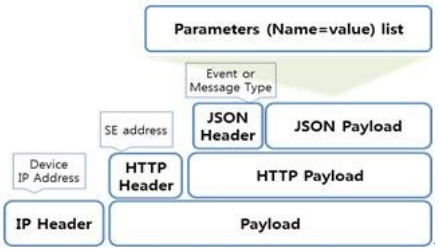


Figure 5. The Structure of RUI Message

The Address of RUI message consists of 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 (RUI Manager, RUI Agents).

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).

C. 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 6.

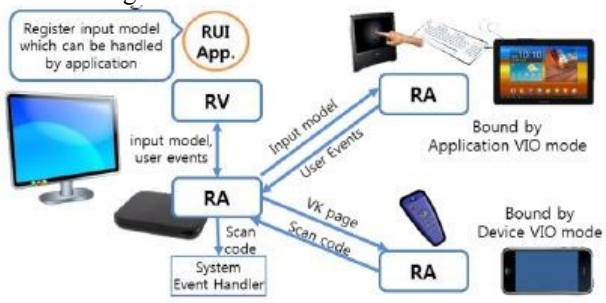


Figure 6. Virtual IO action model

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 RUI Agent 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 RUI Viewers of other devices. The Virtual IO Page is transmitted to other devices and launched by RUI Viewers of the devices, when users want to control remote devices by virtual IO mode. user input events are transmitted to remote RUI Agent, and the user events will be consumed by system event handler.

The second mode is the application virtual IO mode which RUI applications can select necessary type of virtual controller dynamically. Each RUI Agent manages embedded virtual IO emulators to be launched by only local RUI Viewer, not by remote RUI Viewers. Virtual IO emulators are predefined and installed in the repository of the RUI Agent according to the capabilities of local devices.

User device is virtualized into the emulator which can process the event model described in sub-description for each UI segment. 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. Moreover, RUI applications control the layout of virtual IO pages by send RUI messages which contains the event list. user input events are transmitted as user events which will be invoked to RUI application's event queue by the RUI Viewer.

III. REFERENCE IMPLEMENTATION

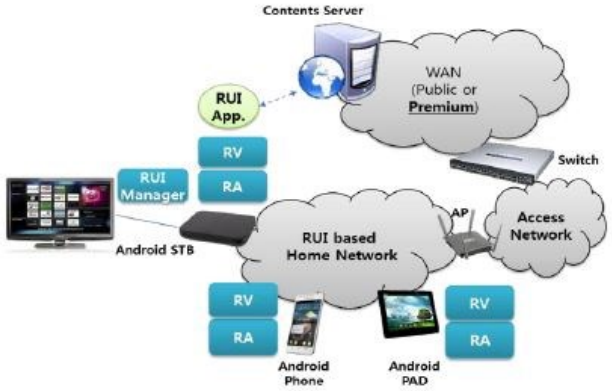


Figure 7. Network configuration of a reference implementation

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 7 and Table 1.

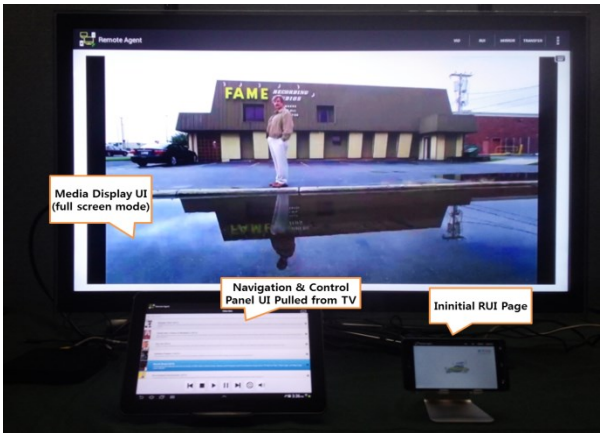
Android based a set-top box; a smart phone and a smart pad are interconnected by an AP connected to a Giga-bit switch. A VOD server is connected directly to switch as an UPnP AV server. VOD client is installed the set-top box as an UPnP AV Renderer. The RUI Manager runs on the smart pad to manage the 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.

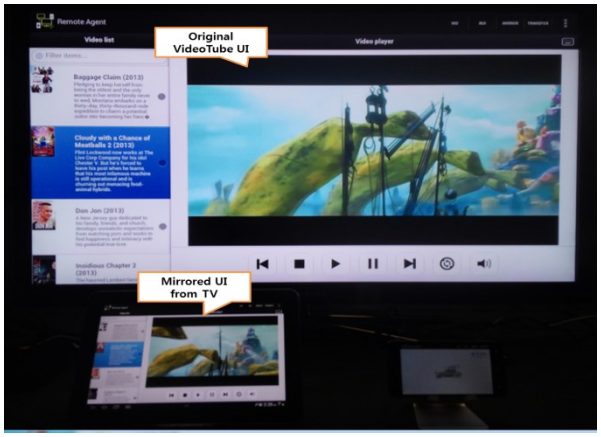
TABLE 1. THE DETAILS OF DEVICE AND SOFTWARE

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) ■ 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

VideoTube RUI service was implemented for remote UI sharing functionality among home-networked devices as Figure 8. VideoTube is composed of a contents server and a media player to provide users with a VOD-like service.



(a) Segmenting RUI mode



(b) Mirroring RUI mode

Figure 8. A Exemplary RUI service (Segmenting & Mirroring)

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 8, 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 $2^{\wedge 3}$ 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 8. 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.

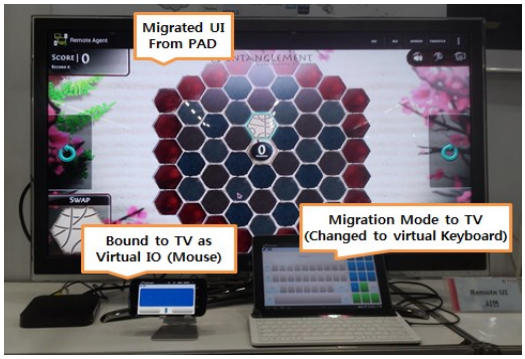


Figure 9. An Exemplary Virtual IO service (Mouse & Keyboard)

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.

An Entanglement game [11] 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. CONCLUSIONS

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.

TABLE 2. THE COMPARISON OF RUI PROTOCOLS

Item	**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 2 shows that the proposed framework is better than other RUI standards for various RUI functionalities.

ACKNOWLEDGMENT

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].

REFERENCES

[1] 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.

[2] Hyori Jeon, Yonghee Shin, Munkee Choi, Jae Jeung Rho, and Myungseuk Kim, "User Adoption Model under Service Competitive Market Structure for Next-Generation Media Services," *ETRI Journal*, vol. 33, No.1, pp.110–120, Feb. 2011.

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

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

[5] Bong-Jin Oh and Jong-Youl Park, "Design and Implementation of HTML5 based Collaborative N-Screen Contents Platform for Smart TV," in *Proc. ICCC2013*, paper 038, pp.225–226.

[6] 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*, paper 7.1(3), pp.192–194.

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

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

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

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

[11] 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.