Enhancing Home Entertainment Experience with Virtual UPnP Media Renderers

Chunglae Cho, Sung-Soo Kim and Kyeong-Deok Moon
IT Convergence Technology Research Laboratory, ETRI
Daejeon, Korea, 305-700
Email: {clcho, sungsoo, kdmoon}@etri.re.kr

***Abstract*—In this paper, we introduce a system called Virtual Device Ensembler (VDE) providing virtual UPnP renderers which are combined with the rendering functions of real UPnP renderers. We present the architecture of the system and discuss the issues raised when we implement the system.**

***Keywords*—Virtual Media Renderer, UPnP, Home Network.**

I. Introduction

Suppose that you want to watch a movie alone at midnight while other family members are sleeping. Then, either you should set the volume very low if you watch it on your TV or you should watch it on your smart pad using your headphone but the screen size of the pad may not be satisfiable. It would be desirable that you watch the movie on the TV with the sound muted and listen to the audio from the headphone connected to your pad. In this case, it is likely that you use a virtual media renderer composed of the TV and the pad serving as a video renderer and an audio renderer, respectively. There are many other use cases where such virtual renderers are useful. For example, you may want to watch a music video on the TV but listen to the audio from the high quality audio player. In this case, it is desirable that you use a virtual media renderer composed of the TV and the audio player.

In this paper, we present a system called Virtual Device Ensembler (VDE) that enables us to build a virtual media renderer by combining multiple UPnP media renderers. Users can virtually assemble the existing UPnP renderers into a virtual renderer. The major advantage of this idea is that it works like a set of real UPnP media renderers. That is, they can select and use the virtual UPnP renderer from their typical UPnP control points just like other real UPnP renderers. Fig. 1 shows how VDE works in the home network consisting of UPnP media servers and renderers. Since VDE is considered as a UPnP media renderer, users can select it as a sink of the AV stream on their control point. Then, when they command the virtual renderer to play the content they select, the VDE receives the original AV stream of the content and it demultiplexes it into a video stream and an audio stream, which are forwarded to the associated real renderers, respectively. Users can continue to use their existing systems without any modification and get a new entertainment experience. They only need to install a new small box (i.e., VDE) and set up their own virtual renderers.

There are some previous work on providing virtual UPnP devices in the home network but none of them presents virtual

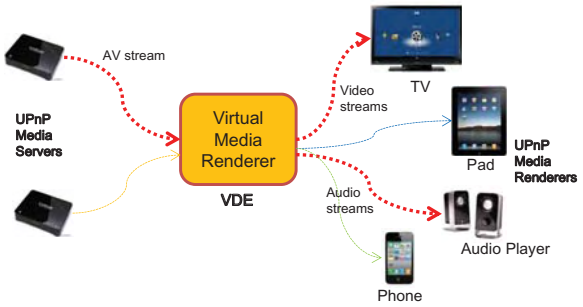

Fig. 1. Service concept of Virtual Device Ensembler.

UPnP renderers. Either they introduce virtual UPnP servers providing unified access to distributed media content, e.g., [1], [2], or present virtual devices for connecting non-UPnP devices to UPnP networks [3], [4].

The rest of the paper is organized as follows. In Sec. II, we present the architecture of VDE and explain how the components of VDE work. We discuss the issues of implementing VDE briefly in Sec. III and give the concluding remark in Sec. IV.

II. Architecture of VDE

In this section, we introduce the architecture of VDE, which is shown in Fig. 2, and explain how it works to realize the scenario introduced in Sec. I. VDE consists of two subsystems: Virtual UPnP Renderer Manager (VURM) and Resource Collaboration Server (RCS). VURM manages the instances of the virtual UPnP media renderers. It accepts all the UPnP search and control requests to the virtual renderers. For example, when it receives a generic search message from a control point, it replies as if it is a set of the UPnP renderers. Device and service descriptions of a virtual renderer are generated based on the descriptions of real renderers. When control messages to a virtual renderer is received, VURM interprets the requests and converts them into proper UPnP and RCS control messages.

The Virtual Resource Dependency Manager (VRDM) maintains the relationship between a virtual renderer and its associated real renderers and manages the life cycle of the virtual UPnP media renderers. VRDM validates the aliveness of the virtual renderers. If an associated real renderer leaves from the network, it changes the state of the virtual renderer to

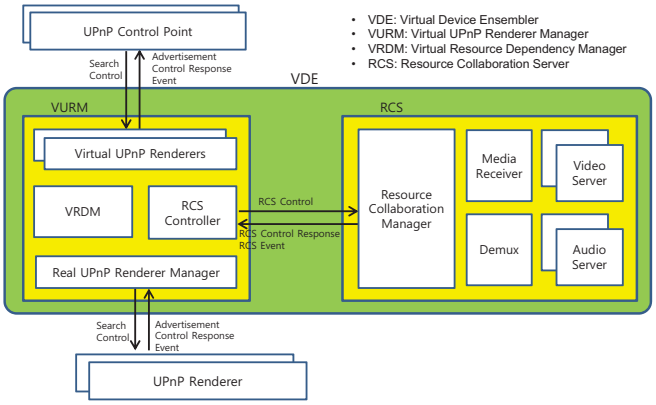

Fig. 2. Architecture of Virtual Device Ensembler.

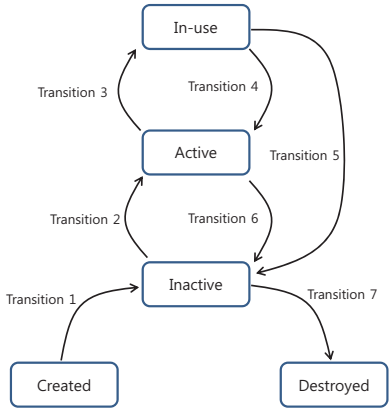

Fig. 3. Life cycle of a virtual renderer.

the inactive state so that users cannot use it. The Real UPnP Renderer Manager is the interface to the real renderers. It maintains the information of alive UPnP renderers. It also sends control messages and receives events from them.

When a user selects a content to play it on a virtual renderer, VURM receives the control message and prepares RCS to start to receive the AV stream and to demultiplex it. RCS plays the roles of the media stream receiver and server. It receives the AV stream of the selected content and demultiplexes it into two streams: video stream and audio stream. Then, it serves as stream servers of the video and the audio. At this moment, VURM controls the associated real UPnP renderers to start to receive the video and audio streams from RCS, not from the original media server.

A virtual renderer has five states in their life cycle as shown in Fig. 3. When a virtual renderer is generated, VRDM sets the virtual renderer to the created state. Then, the virtual renderer moves to the inactive state, which implies that it is registered to VURM. If all the associated real renderers of the virtual renderer are currently alive and ready to use, then it moves to the active state. The active state means that the virtual renderer is ready to use. Then, it advertises itself and UPnP control points can search it. If the virtual renderer is being used by a user, it moves to the in-use state. It implies that another control point is prohibited from using it. When the virtual renderer is in the active or in-use states, it moves back to the inactive state if some of the associated real renderers are not alive or not available. When the virtual renderer is destroyed by a user, it moves to the destroyed state, which implies that it is not managed by the system any more.

III. Issues

In this section, we discuss some issues on implementing the VDE system. First, playing content on a virtual renderer associated with multiple real renderers raises the issue of synchronized playback. The real renderers should start rendering the content synchronously. Even if the VDE sends the play messages to the associated real renderers at the same time, it is very unlikely that they start to play at the same time since they may have different buffering delay. UPnP AV architecture version 2 defines the *CLOCKSYNC feature* enabling synchronized playback [5]. It shows how to configure and manage synchronized playback of multiple independent pieces of matched content. Using the feature, VDE can set the real renderers to start rendering the content at the requested presentation time. Second, it is required that VDE support capability matching between media servers and real renderers. The capability matching includes the content format matching and transport protocol matching. The capability of a virtual renderer should not have to be restricted by the capabilities of the associated real renderers. For example, the virtual renderers may advertise more content formats than those supported by the associated real renderers if VDE can perform the media format transformation. Similarly, suppose that the media server only supports RTP-RTSP but the real renderers only support HTTP. Even in this situation, it is possible that VDE receives the content from the server using RTP-RTSP protocol and serves the transformed content to the real renderers using HTTP.

IV. Conclusion

In this paper, we have presented the VDE system to provide virtual UPnP renderers. Users can create virtual UPnP renderers by combining the rendering functions of real UPnP renderers. Each associated real renderer functions as either AV, video or audio renderer. Users can extend their entertainment experience by building their own virtual renderers.

References

[1] D. Kang, K. Kang, S. Choi, and J. Lee, "UPnP AV architectural multimedia system with a home gateway powered by the OSGi platform," *IEEE Transactions on Consumer Electronics*, vol. 51, no. 1, pp. 87–93, 2005.

[2] J. Park and S. Kim, "A transparent contents sharing service with virtual media server," in *International Conference on Convergence Information Technology*, 2007, pp. 764–767.

[3] H. Park, M. Choi, E. Paik, and N. Kim, "Interoperability model for devices over heterogeneous home networks," *IEEE Transactions on Consumer Electronics*, vol. 55, no. 3, pp. 1185–1191, 2009.

[4] J. Sung, T. Kim, and D. Kim, "Integration of IEEE1451 sensor networks and UPnP," in *IEEE Consumer Communications and Networking Conference (CCNC)*, 2010, pp. 1–2.

[5] *UPnP AV Architecture Version 2*, UPnP Forum, December 2010, http://www.upnp.org/.