Software Defined Radio Controller Using Bluetooth

Hyoseok Yoon
Contents Convergence Research Center
Korea Electronics Technology Institute
Seoul, Korea
hyoon@keti.re.kr

Saet-Byeol Yu
Contents Convergence Research Center
Korea Electronics Technology Institute
Seoul, Korea
sbyu@keti.re.kr

Se-Ho Park
Contents Convergence Research Center
Korea Electronics Technology Institute
Seoul, Korea
sehopark@keti.re.kr

***Abstract*—**We present a Bluetooth-based mobile application to support and enhance internal and external interfaces to software defined radio platforms. For this application, a control message protocol used between software defined radio platform and smart devices is designed and implemented with examples of various control flows. With enhanced controllability and smart device connectivity through Bluetooth, versatile applications such as connected audio applications and in-vehicle infotainment systems can be realized.

Keywords—radio control; software radio; bluetooth; mobile applications; networked control systems

I. Introduction

Software Defined Radio (SDR) provides flexible and reconfigurable architecture [1] that can be used to provide context-aware services in cognitive radio domain [2] and enhance the degree of programmability of network devices [3]. Current states of the art and technical challenges for SDR are addressed in [4] [5] [6]. In this paper, we propose and implement a Bluetooth-based mobile controller for SDR, named SDRController, to support interfaces between SDR hardware and smart devices [7] to enable novel applications such as in-vehicle infotainment systems [8]. To show feasibility of our approach, a control message protocol and control flows are demonstrated with the proposed SDRController.

II. Smart Convergence Platform

A Smart Convergence Platform (SCP) aims to receive multi-standards radio, control and reconfigure based on SDR architecture, and communicate with smart devices via Bluetooth. Our initial attempts on its preliminary concept, design and implemented simulation are described in [7]. Fig. 1 shows a refined SCP that integrates a Software Defined Radio Platform (SDRP) and a Connected Audio Application Platform (CAAP). The SDRP's role is to receive and control multi-standards radio within the SCP, whereas the CAAP's role is to interface the SCP to external smart devices.

With a mobile controller application named SDRController, user-requested messages are first delivered to the CAAP using Bluetooth Serial Port Profile (SPP). Then the CAAP relays the received messages to the SDRP using a universal asynchronous receiver-transmitter (UART). Finally, the SDRP is controlled by messages received from the CAAP. For transmitting audio data, I^2C is used between the SDRP and the CAAP, while Bluetooth Advanced Audio Distribution Profile (A2DP) is used between the CAAP and a connected Bluetooth speaker. An interaction flow among the SCP, the SDRPController, and the speaker is depicted in Fig. 2.

This research was supported by the Ministry of Trade, Industry and Energy (MOTIE) and Korea Evaluation Institute of Industrial Technology (KEIT). [10063285, Development of RF SoC and Smart Convergence Platform for SDR (Software Defined Radio)]

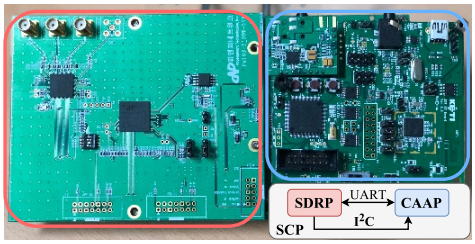

Fig. 1. An integrated SCP encompassing the SDRP and the CAAP.

Fig. 2. An interaction flow for using the SCP.

A. Control Message Protocol

We designed and implemented a consistent protocol for control messages for both types of internal (i.e., system-level) and external interfaces for multi-standards radio. For example, when the SDRP starts initialization steps, system-level control messages and Digital Audio Broadcasting (DAB) initialization messages are delivered following the same protocol.

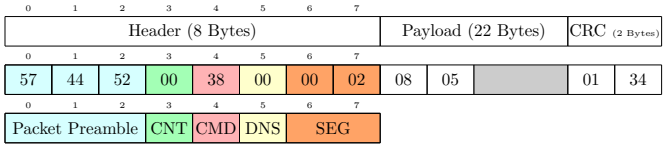

Fig. 3. The designed control message protocol, an exemplar message, and detailed header description (CNT = Counter, CMD = Command ID, DNS = Device & Standard IDs, SEG = Segmentation).

Fig. 3 represents a 32 bytes packet protocol for control messages that is composed of 8 bytes of a header, 22 bytes of a payload, and 2 bytes of a cyclic redundancy check (CRC), respectively. Specifically, the header is composed of 3 bytes for

packet preamble, 1 byte for counter, 1 byte for command ID, 1 byte for device & standard IDs, and 2 bytes for a segmentation flag and length.

B. Examples of Control Message Flow

To demonstrate interactions between the CAAP who interfaces control messages with the SDRController and the SDRP, we present 3 representative control message flows.

1) Initialization of the SDRP: Fig. 4 shows a control message flow between the CAAP and the SDRP for the initialization. When control messages from the CAAP are successfully received and processed in the SDRP, the SDRP sends back corresponding response messages. Consequently, the CAAP is notified with an execution state and results.

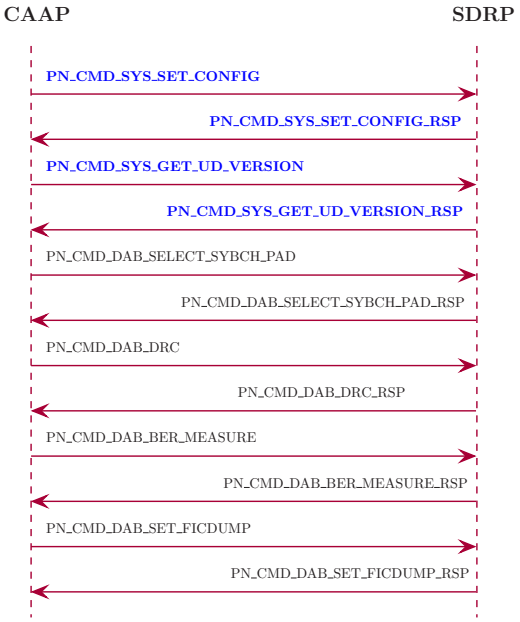

Fig. 4. A sequence diagram for the SDRP initialization where system-level control messages are highlighted in blue.

2) Channel Scan: A sequence diagram for a channel scan is shown in Fig. 5. A channel scan with specified frequency ranges is issued with a `PN_CMD_DAB_TUNE` message. On the SDRP, a response message is sent back to the CAAP, followed by messages that contain service information (ensemble, service, and service component information) and a `PN_NOTY_DAB_SCANINFO` on completion. Consequently, the CAAP parses and retrieves service information.

3) Channel Selection: After completing channel scan, a `PN_CMD_DAB_SELECT` control message with appropriate service parameters is delivered to the SDRP for channel selection as shown in Fig. 6. According to the service parameters, an intended frequency is tuned and corresponding service is selected for playing a connected audio service. A successful channel selection from the CAAP is acknowledged by a response message from the SDRP. To change or stop the current service, a `PN_CMD_DAB_ACTSTOP` message is used.

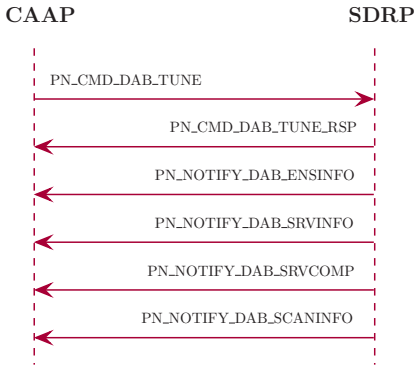

Fig. 5. A sequence diagram for channel scan.

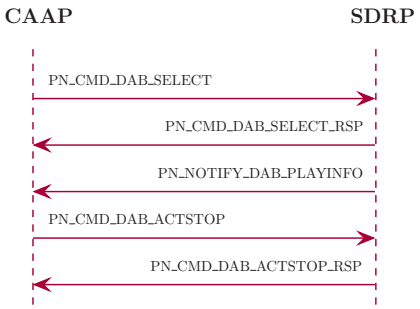

Fig. 6. A sequence diagram for channel selection.

III. Implementation of SDRController

To conveniently control the SCP, we implemented a Bluetooth-based mobile application (SDRController) that can communicate with the CAAP for controlling the SDRP. Our implementation is based on Android OS and we used Bluetooth SPP to control the CAAP. By using SPP, a virtual yet working serial communication between Bluetooth devices is implemented. Following the specification of SPP and roles, the CAAP connected to the SDRP is implemented as an 'Acceptor' that awaits other devices to connect and the SDRController is implemented as an 'Initiator' that connects to the CAAP. Fig. 7 shows our testbed.

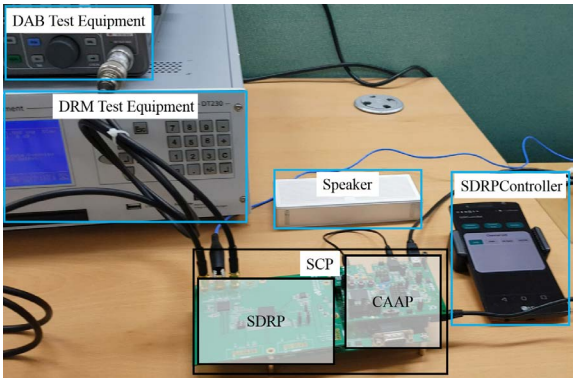

Fig. 7. Our testbed includes a DAB test equipment, a DRM test equipment, the SCP consisted of the SDRP and the CAAP, an Android-based smartphone and a Bluetooth speaker.

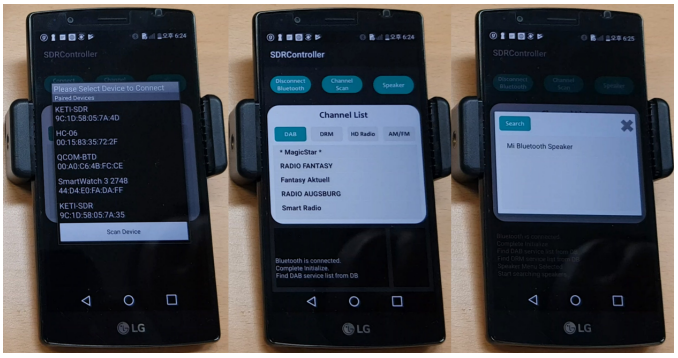

Fig. 8. The SDRController showing results of `Connect Bluetooth`, `Channel Scan`, and `Speaker` commands, respectively.

The GUI of SDRController is divided into three areas as shown in Fig. 8. The upper area is designated for buttons that execute functional commands of `Connect Bluetooth`, `Channel Scan`, and `Speaker`. The central area is used to display channel lists of various radio standards, i.e., DAB, DRM, HD Radio, and AM/FM. The bottom area is reserved to display logs for development purposes and current status of the SCP. Results of executing three commands are depicted in Fig. 8 and detailed processes are described.

A. `Connect Bluetooth` Command

When a user clicks on `Connect Bluetooth` button, a displayed pop-up window allows the user to search, connect, and configure a CAAP. Selecting a CAAP from the displayed list establishes a Bluetooth SPP connection between the SDRController and the CAAP. Then initialization control messages are sent to the CAAP. The CAAP relays control messages received from the SDRController to the SDRP via UART. Then the SDRP is initialized according to the control messages and response messages are returned back to the CAAP via UART. Finally, the CAAP relays the response messages to the SDRController through the Bluetooth SPP.

B. `Channel Scan` Command

When `Channel Scan` button is clicked on the SDRController, the SDRController sends channel scan control messages to the CAAP who relays the control messages to the SDRP via UART to perform a full channel scan. For example, 15 frequencies are fully scanned for the DAB standard which takes considerable amount of time. After completing the full scan, channel or service names are displayed on the central area of GUI. By configuring a service filter, we only display audio and video broadcasting services. To reduce unnecessary burden of repetitive channel scanning, retrieved channel information is stored on an on-device application database after the first full scan. Therefore launching the SDRController and establishing Bluetooth connection automatically display stored channel information by accessing the on-device application database. A `Channel Scan` to the SDRP is responded with service information. Then channel information and other required information for tuning (Fig. 9) are saved to the DB with a unique `SERVICE_ID` as a key.

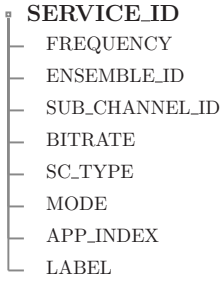

Fig. 9. Channel information saved on the application DB.

C. `Speaker` Command

This command invokes the CAAP to search and connect for Bluetooth devices capable of playing audio through A2DP. The CAAP uses a Class of Device (CoD) filter of `0x04` to search audio devices only. Two standards of radio broadcasting service, DAB and DRM are supported. For each radio service selected, a scanned channel service list is displayed where the user can select a service to play through the speaker.

IV. Conclusion

In this paper, we demonstrated how our proposed SDRController and its control message protocol and control flows can be used as a tool for the SCP, enhancing internal and external interfaces as well as controllability of the SDRP using Bluetooth.

Acknowledgment

Authors thank PnpNetwork Technologies, Inc. for development, integration, and evaluation of the SDRP and the SCP.

References

[1] U. Ramacher, "Software-defined radio prospects for multistandard mobile phones," *Computer*, vol. 40, no. 10, pp. 62–69, Oct 2007.

[2] T. Ulversoy, "Software defined radio: Challenges and opportunities," *IEEE Communications Surveys Tutorials*, vol. 12, no. 4, pp. 531–550, Fourth 2010.

[3] D. F. Macedo, D. Guedes, L. F. M. Vieira, M. A. M. Vieira, and M. Nogueira, "Programmable networks—from software-defined radio to softwareo software-defined networking," *IEEE Communications Surveys Tutorials*, vol. 17, no. 2, pp. 1102–1125, Secondquarter 2015.

[4] G. Sklivanitis, A. Gannon, S. N. Batalama, and D. A. Pados, "Addressing next-generation wireless challenges with commercial software-defined radio platforms," *IEEE Communications Magazine*, vol. 54, no. 1, pp. 59–67, January 2016.

[5] A. M. Wyglinski, D. P. Orofino, M. N. Ettus, and T. W. Rondeau, "Revolutionizing software defined radio: case studies in hardware, software, and education," *IEEE Communications Magazine*, vol. 54, no. 1, pp. 68–75, January 2016.

[6] X. Wei, H. Liu, Z. Geng, K. Zheng, R. Xu, Y. Liu, and P. Chen, "Software defined radio implementation of a non-orthogonal multiple access system towards 5g," *IEEE Access*, vol. 4, pp. 9604–9613, 2016.

[7] H. Yoon, J.-E. Lee, S.-B. Yu, and S.-H. Park, "Bluetooth-enabled software defined radio platform," *International Journal of Future Generation Communication and Networking*, vol. 10, no. 7, pp. 1–12, Jul. 2017.

[8] H. Yoon and S.-H. Park, "Enabling in-vehicle infotainment with bluetooth connectivity," *Journal of Industrial Information Technology and Application*, vol. 1, no. 1, pp. 12–18, Jun. 2017.