Telfor Journal, Vol. 5, No. 1, 2013.

An Approach of Integrating Communication Services in Applications for Android-Based Digital TV Receivers

Stefan Jovanovic, Marija Punt, *Member, IEEE,* Milan Z. Bjelica *Member, IEEE,* Vladan Zdravkovic, and Majda Kukolj

***Abstract* — Digital TV receivers are becoming increasingly powerful devices allowing consumers to not only watch television broadcasts but also to access the Internet or communicate to other devices in the same local area network through either an Ethernet connection or by using a wireless connection. As the living room represents a meeting place for family and friends to gather and socialize in, the possibility of playing informal games using the television set as the interaction device is very attractive. This paper presents a developed application that integrates new communication capabilities of digital TV receivers running the Android OS. The application is a game showing its content overlaid on top of a television program whereas Android mobile devices are used as controllers. The performance of the application is tested by measuring the response times of the various communication services and by analyzing feedback from a selected group of users.**

***Keywords* — Android OS, digital TV receiver, mobile devices, TV-centric applications, games.**

I. Introduction

A digital TV receiver or set-top box is a device that is able to receive and decode a Digital Video Broadcasting (DVB) signal. Lately digital TV receivers are becoming more and more powerful, with the possibility of running different applications, while having access to the Internet, so that users can, besides watching television, check their e-mail, look at the weather forecast, see their favorite videos or interact with their friends on social networks from their living rooms. Considering the fact that digital TV receivers are most often part of a local area network, it enables them to communicate with other devices in the same local area network. As the living room represents a focal point for family and friends to gather and socialize in, the possibility of playing informal games where the television and mobile devices can be used for interaction is very appealing [1].

The Android operating system has become a leading open platform for the development of applications for mobile devices (smart-phones and tablets), and, most recently, digital TV receivers (based on Google TV) [2]. The research in [3] proposes an integration of digital TV (DTV) services in Android-based digital TV receivers allowing Android applications to benefit from broadcast related data.

This work was partially supported by the Ministry of Education and Science of the Republic of Serbia, project No. 32041, and 44009.

Stefan Jovanovic is with the School of Electrical Engineering, University of Belgrade, Bulevar kralja Aleksandra 73, 11120 Belgrade, Serbia (e-mail: js103255m@student.etf.bg.ac.rs).

Marija Punt is with the School of Electrical Engineering, University of Belgrade, Bulevar kralja Aleksandra 73, 11120 Belgrade, Serbia (phone: 381-11-3218392, e-mail: marija.punt@etf.bg.ac.rs).

Milan Z. Bjelica, is with the Computer Engineering and Computer Communications Department, Faculty of Technical Sciences, University of Novi Sad, Trg Dositeja Obradovića 6, 21000 Novi Sad, Serbia (phone: 381-21-4801204, e-mail: milan.bjelica@uns.ac.rs).

Vladan Zdravkovic is with the Sheffield Hallam University, Howard Street, Sheffield, South Yorkshire S1 1WB, United Kingdom; (e-mail: vladanx@gmail.com).

Majda Kukolj is with the RT-RK Computer Based Systems LLC, Narodnog Fronta 23a, 21000 Novi Sad; Serbia (telefon: 381-21-4801247, e-mail: majda.kukolj@rt-rk.com).

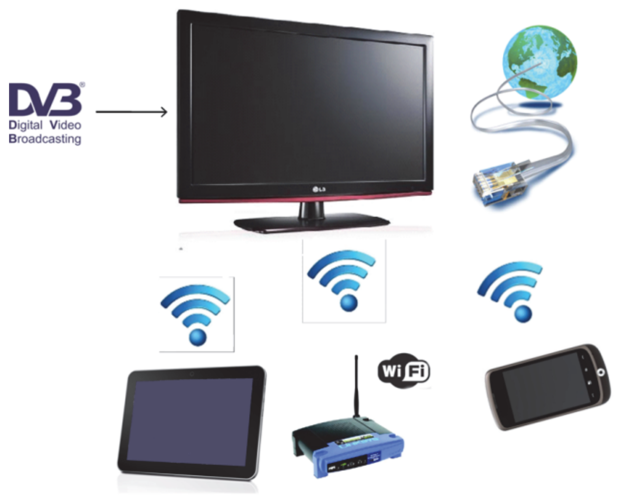

Fig. 1. Example environment used by the *Egg and Spoon* application.

This paper presents an application implemented using the Android DTV Java service in order to examine the possibilities of integrating access to the DTV services and features of digital TV receivers, applying them in an environment such as the living room. A client-server TV-centric game called *Egg and Spoon* was selected as a representative example. The game is played during TV shows on a graphical layer on top of the broadcasted television stream, the game processes data coming from the DVB stream such as the electronic program guide, and communicates with external web servers by sending game results to popular social networks. Mobile devices that are connected to the digital TV receivers in the local area network are used as game controllers by employing the built-in touch-screens and motion sensors. An example of the environment in which the *Egg and Spoon* application is developed is shown in Fig. 1.

The rest of the paper is organized as follows. Section II deals with trends in game development for television and mobile devices. Section III describes the realization of the *Egg and Spoon* game. The Android DTV Java service used to access the DVB stream is presented in Section IV. Section V shows application performance test results by measuring the response times for the various communication protocols as well as an analysis of user experiences. Section VI presents the conclusions.

II. Trends in Games for Television and Mobile Devices

The latest research shows that TVs are beginning to play an important role in social communication and that activities such as playing informal games with family members, with the TV as a medium for interaction, is becoming more and more appealing [4]. An example of such an application, which combines game characteristics with interactive qualities of the television is *Wize* [5]. This application allows the TV viewer to participate in an interactive quiz while watching the show.

Mobile devices such as smart phones and tablets are equipped with touch screens and motion sensors allowing novel user interactions in games. Using these features, game developers have new forms of exploring user input, necessary to adapt or create new kinds of game play. An example of such a game is a first-person driving game titled *Tunnel Run* where instead of controlling a car through the pressing of buttons a "tilt" interface was used [6]. Another two examples are fishing and bowling game controlled by using only motion-sequence recognition [7]. Apart from being used to play games, the latest research suggests the use of mobile phones as TV remote controls [8].

A TV-centric board game where the main game board is shown on TV and the private content is shown on mobile devices is presented in [9]. The *Egg and Spoon* game uses a similar idea, extending it with the ability to incorporate both DTV functionality and Internet access into the game application.

III. The Egg and Spoon Application

The developed game represents a speed race between two players, each of whom is carrying one egg in a spoon. The race is finished when one of the players reaches the goal first with the egg in the spoon, or at the moment one of the racers drops the egg. The moment the race is finished the users can leave the game or start a new round.

A. Description of the Application

The client application running on a mobile device automatically scans the network and finds the servers that are hosting the game. If the local area network offers multiple servers, the user is shown a pop-up menu with a list of all available servers. Clicking on one of them starts the connection with the server. Each connected client controls one racer in the game. The player propels the racer toward the finish line by rapidly pressing the left and right buttons in alternating order, simultaneously the player must prevent the egg falling off the spoon displayed on the screen by balancing the mobile device. A screenshot of the client-side application during game play is shown in Fig. 2.

On the server application that is running on the set-top box, the TV program is shown, and the user can perform all of the activities typical for a modern TV, including viewing of the electronic program guide or teletext, choosing an appropriate channel and recording it.

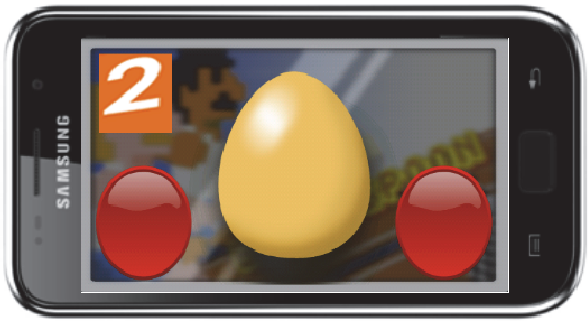

Fig. 2. Screenshot of the *Egg and Spoon* controller user interface.

In the background the server is listening to the designated port and waiting for potential clients to connect. When both clients are connected, the server assigns them corresponding parameters and informs them the game is about to start. During the game the server receives the messages sent by the clients, updates the status of the game and based on the current state of the game renders the graphics on the screen. Once the game is finished the user chooses whether to publish the results on the Twitter social network (e.g. "John has swept away Mike while watching show A on channel B"). The game is shown on top of the TV program, taking only a minimal part of the screen. A screenshot of the server-side application during game play is shown in Fig. 3.

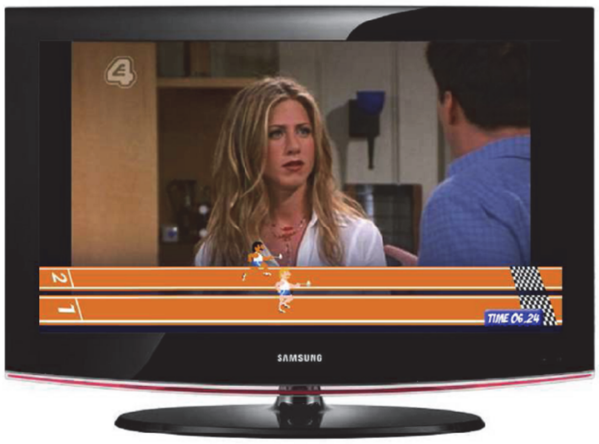

Fig. 3. Screenshot of the *Egg and Spoon* server-side user interface.

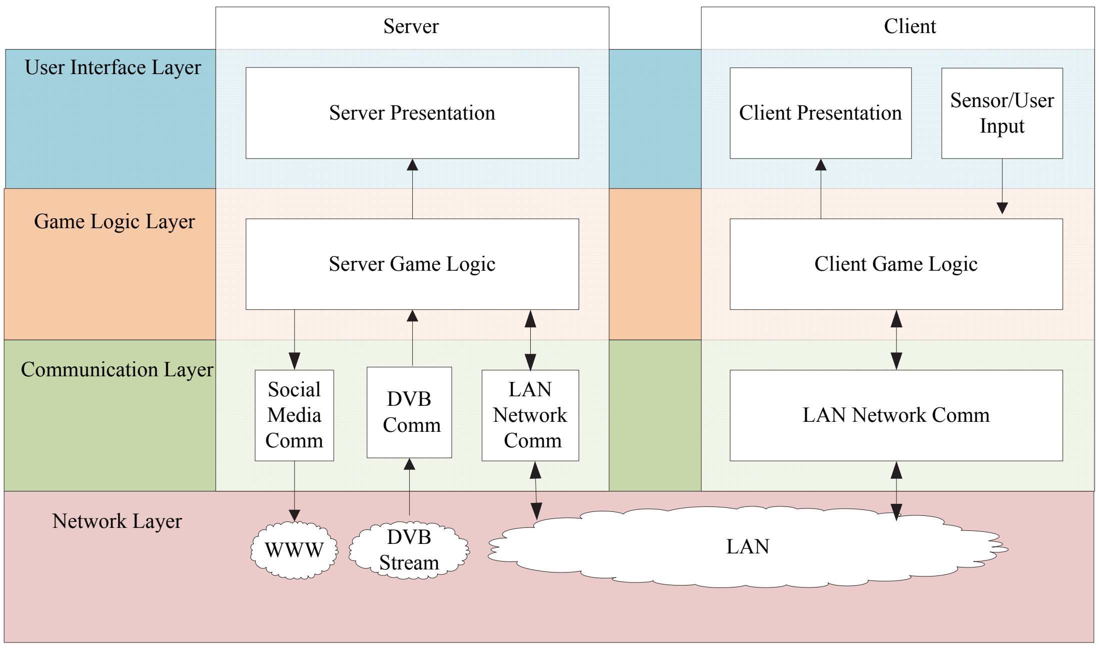

Fig. 4. The architecture of the *Egg and Spoon* application.

After the game has finished, the system logs into its Twitter account and publishes the results of the game along with a screenshot. The results are published only if both players have confirmed their consent. The screenshot is taken at the precise moment of the race finish, and it is clearly evident who won, in what time and how convincingly. The format and text of the message depend on the time in which the race was over, the players' duel, and the best achieved results.

B. Architecture of the application

The client and server-side are realized in three independent layers: Communication, Game Logic, and User Interface. The architecture of the application is shown in Fig. 4.

The Communication layer on the server-side contains three components: *Social Media Comm*, *LAN Network Comm* and *DVB Comm* components, and on the client-side it only contains the *LAN Network Comm* component. The *Server LAN Network Comm* component can maintain a larger number of TCP/IP connections with clients. The *Client LAN Network Comm* component implements complementary logic, it is in charge of initiating the connection by finding the server and creating a connection towards one server only. Both server and client can send XML messages to each other using the communication layer. Each message consists of three fields: subject, value and source. The subject field explains the type of the message that is sent. The value field contains a parameter value relevant to the type of message specified by the subject field. The source field identifies the message sender. The message shown in Fig. 5. is an example of player 1 sending a message to the server indicating the speed of the runner.

The *Server DVB Comm* component enables access to information from the DVB stream by using the Android DTV Java service [10]. The *Social Media Comm* component is in charge of the connection with an external web server (in this case, Twitter) through the HTTP protocol.

```
<message>
    <subject>SPEED</subject>
    <value>20</value>
    <source>PLAYER_1</source>
</message>
```

Fig. 5. Example of player 1 sending a XML message to the server indicating the speed of the runner.

The Game Logic layer on the server-side consists of the *Server Game Logic* component, and on the client-side it consists of the *Client Game Logic* component. The *Server Game Logic* component implements a state machine. The state is updated within a loop, and every loop iteration is executed in a fixed time interval. The state change is influenced by messages received from the clients through the communication layer. On the client-side a mini-game is implemented as a state machine and its state is influenced by information from the sensors, user input or incoming messages from the server (e.g. regarding the end of the game).

The User Interface on the server-side consists of the *Server Presentation* component, and on the client-side it consists of the *Client Presentation* and *Sensor/User Input* components. The *Server Presentation* component is in charge of rendering the game animations on top of the video layer received through a video decoder on the set-top box. The game graphics are resized to fit the size of the screen. The *Client Presentation* component is responsible for rendering the graphics on the mobile device, showing an animation of an egg falling either to the left or to the right. A part of the screen contains virtual buttons used to control the speed of the racer. The *Sensor/Input User* component processes the input from the virtual buttons, as well as collects and processes data from the sensors. On the client-side the entire screen surface is used to display the game.

IV. Android DTV Java Service

On the server-side the *Egg and Spoon* is coupled with the digital television middleware providing all required information about the DVB transport stream and current state of the digital TV receiver. For example, it is possible to obtain information about the currently watched service (such as channels) and the expected program schedule for all the services in the service list. Additionally, it is possible to issue commands to a digital TV receiver, such as switching channels or volume control. The position of the *Egg and Spoon* application in relation to the DTV architecture is shown in Fig. 6.

The DTV functionality is exposed to the *Egg and Spoon* through the Android DTV Java service. This DTV service is initialized upon the startup of the set-top box. The *Egg and Spoon* application communicates with the service using the Android Binder mechanisms, to obtain an instance to a class implementing the *IDTVManager* interface, which gives access to the DTV API. Through the *IDTVManager,* the *Egg and Spoon* application has the ability to fetch a specific interface towards a desired piece of DTV functionality. The class diagram of available interfaces is given in Fig. 7.

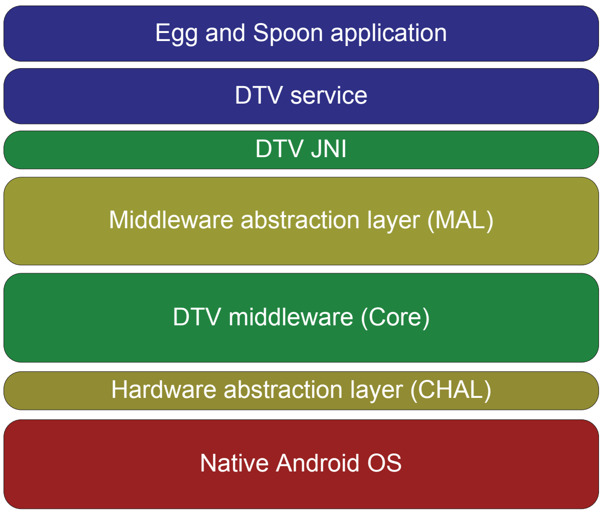
Egg and Spoon application
DTV service
DTV JNI
Middleware abstraction layer (MAL)
DTV middleware (Core)
Hardware abstraction layer (CHAL)
Native Android OS

Fig. 6. *Egg and Spoon* application in relation to the DTV service architecture.

The DTV service is coupled with the DTV middleware running as a native Android application (written in C). This coupling is performed through Java Native Interface (JNI) mechanisms, which enable Java-based services to execute routines written in C. On the other hand, all the events generated by the DTV middleware, are reported back to the DTV service through JNI, by calling a Java method designated for the event reception. This method parses the event report and passes it on to an appropriate DTV Java object.

The JNI implementation accesses the DTV middleware core through a middleware abstraction layer API. The middleware abstraction layer provides a unique API for all available DTV functionalities. This API is rarely altered while developing software for set-top box devices, making it suitable for implementation-agnostic applications. The DTV middleware core is in charge of monitoring the transport stream, extracting specific service information tables and providing easy-to-access information such as service lists, and event schedules. The middleware core also has access to specific hardware components within a set-top box, therefore allowing clients to tune to a specific frequency, set up processing routes, trigger the recording process, etc. The DTV hardware components present in the set-top box are abstracted from the middleware by a dedicated hardware abstraction layer API.

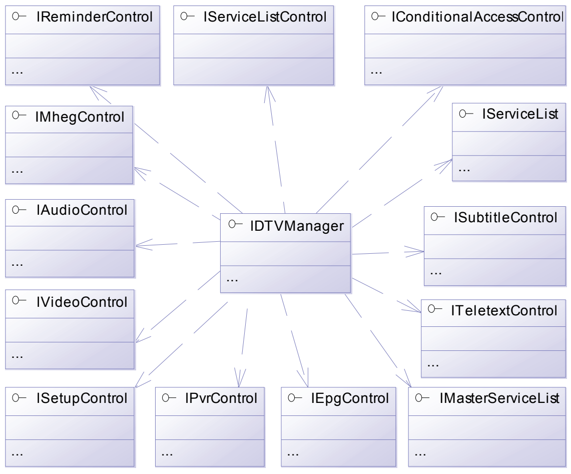

Fig. 7. The available interfaces within the DTV service API.

V. Application Test Results

Measuring the application's performance was done by recording the client-server response time within the local area network, the time needed to render the game graphics on the server-side, the time needed to retrieve information from the DVB stream, and the response time between the application and the Twitter social network.

The server response time was measured in the following manner: the clients sent XML messages to the server, the server received them, unpacked them and then sent the response back to the client. Three different mobile devices were chosen to run the client application. The hardware specifications (processing power, memory and version of the Android OS) were different for each device. Each client sent 10000 individual messages and waited for a reply for each message. For each device the server response time was placed in one of four categories: 0-25 milliseconds, 25-50 milliseconds, 50-75 milliseconds, and more than 75 milliseconds. The distribution of response times over the four categories is presented in Fig. 8. with the x-axis representing different mobile devices, and the y-axis showing the percentage of messages falling in each category in relation to the total number of messages sent. The average measured response time was around 50 milliseconds. If we consider that response values for user gestures under 100 milliseconds appear to the user as instantaneous [11], the server response time may be regarded as acceptable. The time needed for rendering the graphics of the game on the server-side was measured between 11 and 15 milliseconds, which exceeds the target minimum frame rate of 30 frames per second for video games [12].

In order to receive the information about which show is being watched and on what channel, it was necessary to access the EPG (Electronic Program Guide) information and the SDT (Service Description Table) information from the DVB stream. The time necessary for retrieving information about which show is being watched (from the EPG), was measured between 20-30 milliseconds. The time necessary for retrieving information about what channel is being watched (from the SDT) was 1 second. The information is obtained during switching of channels, hence obtaining this information does not add time to sending messages to Twitter.

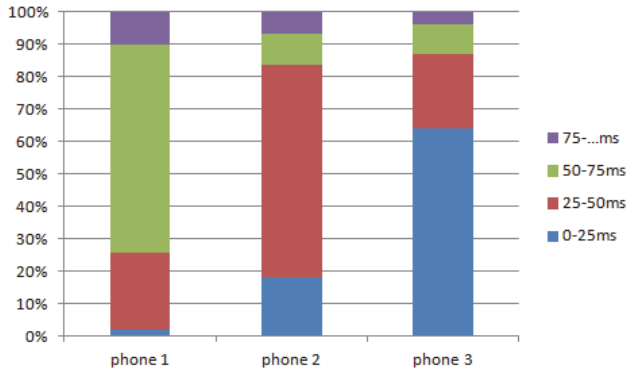

Fig. 8. Server response time distribution per tested device.

Testing the response time of the social network from the application was conducted by sending appropriate messages from the server to the Twitter network, and receiving identifiers as a response, indicating whether the operation was successful or not. The time was measured from the moment the message was sent to the server, up until a response was received. There were no cases in which the server failed to reply. What represents a potential problem is the response time, which often varies. In some cases the reply arrived in only 2 seconds, and there were cases where it took up to 12 seconds to arrive. This delay in communication cannot be influenced in any way by the *Egg and Spoon* application, since the delay is caused by remote servers, belonging to the social network the application is communicating with. The responsiveness of those remote servers varies depending on the load that the social network is under at various times.

When testing user experiences, each participant was explained the concept of the game and instructed on the manner in which the game should be played. After that, every participant tested the game for at least four sessions.

The users had positive experiences regarding response time to user gestures. Besides the new concept of TV-centric games, which most of the participants thought was revolutionary; they also stated that another good aspect was using mobile devices as controllers.

VI. Conclusion

This paper presents an approach to integrate communication services on a digital TV receiver, using the example of the *Egg and Spoon* application as a new type of game which is executed on a set-top box, and controlled by mobile devices. The performance measurements of the responsiveness of the communication protocols and the results of the user analysis show that this game concept is very appealing and acceptable and that the required communication protocol response time satisfies the user needs. Further work would be focused on the development and performance testing of different types of games for the set-top box and mobile devices which would support a larger number of players and thus place a heavier load on the network.

References

[1] M. Z. Bjelica, V. Zdravkovic, M. Punt, and N. Teslic, “TV-centric Gaming Applications for Android OS: Architecture and a Framework”, *Proceedings of the IEEE International Conference on Consumer Electronics (ICCE ’13)*, January 2013.
[2] Google TV, www.google.com/tv/, 2012.
[3] G M. Vidakovic, N. Teslic, T. Maruna and V. Mihic, "Android4TV: a Proposition for Integration of DTV in Android Devices," *IEEE International conference on Consumer Electronics,* 2012.
[4] P. Cesar, D. Geerts, “Past, Present, and Future of social TV: A Categorization”, *Proceedings of the IEEE Consumer Communications and Networking Conference*, pp.347-351, 2011.
[5] P. Almeida, J. Ferraz, A. Pinho, D. Costa "Engaging viewers through social TV games", *Proceedings of the 10th European conference on Interactive tv and video,* pp. 175-184, 2012.
[6] P. Gilbertson, P. Coulton, F. Chehimi, F, and T. Vajk, “ Using ‘tilt’ as an interface to control ‘no-button’ 3-D mobile games”, *Computers in Entertainment (CIE)*, Vol. 6, No. 3, Article No. 38, pp. 1-13, 2008.
[7] J. Baek, B. Yun "A sequence-action recognition applying state machine for user interface", *IEEE Transactions on Consumer Electronics*, pp.719-726, 2008.
[8] C.L. Lin, Y.H. Hung, H.Y. Chen, S.L. Chu "Content-aware smart remote control for Android-based TV", *IEEE International Conference on Consumer Electronics*, pp.678 - 679, 2012 .
[9] N. Kosutic, M. Mitrovic, M. Z. Bjelica, V. Jelovac, ”The Concept and Implementation of a Game Development Platform Based on the Integration of Consumer Electronic Devices with Android OS”, *Proceedings of the ETRAN 2012*, pp. 1-4, 2012.
[10] M. Vidakovic, T. Maruna, N. Teslic, and V. Mihic, “A Java API Interface for the Integration of DTV Services in Embedded Multimedia Devices”, *IEEE Transactions on Consumer Electronics*, Vol. 58, No. 3, pp. 1063–1069, August 2012.
[11] J. Nielsen, “Usability Engineering”, published by Morgan Kaufmann, San Francisco, pp. 134-137, 1993.
[12] M. Claypool, K. Claypool, and F. Damaa. “The Effects of Frame Rate and Resolution on Users Playing First Person Shooter Games”, *Proceedings of the ACM/SPIE Multimedia Computing and Networking (MMCN) Conference,* San Jose, CA, USA, January 2006.