A Foundation for Local Multiplayer Gaming Using Smartphones as Wireless Controllers

By Abule Augustine Arumadri 215014379 15/U/2633/PS

1 Abstract

The purpose of this theory is to give an in-depth technological analysis of a multiplayer local video gaming system designed to let players use their smartphones as controllers, rather than the traditional gaming controllers. The system consists of two interconnected parts: a program (video game) running on a personal computer that acts as a game console, and a smartphone application that turns phones into game controllers, enabling the user to interact with the game in new, creative and exciting ways.

2 Introduction

In today's modern society most people have access to a smartphone [1]. While many use their smartphones for introverted activities such as surfing the web or messaging distant friends, surprisingly few applications allow individuals to use their phone to engage with others in the same physical space.

This situation is problematic for several reasons. Perhaps most noticeably, smartphones can serve as an excuse not to participate in discussions and other types of face to face interaction. A worrying example is the interaction and responsiveness between parents and their children, which might suffer due to excessive smartphone usage [2]. We aim to improve upon the situation by creating a virtual gaming system, where smartphones are used as the controllers and a computer acts as the console. This will allow smartphones to be used for social activities and might even strengthen the bonds of parents and children if they play together.

2.1 Muse and enthusiasm

There are already products on the market that provide similar ways of controlling video games such as the Nintendo Wii U gamepad, Xbox SmartGlass (a companion application for the Xbox 360 and Xbox One video game consoles. Xbox SmartGlass allows players to control console applications using a smartphone or tablet).

Unlike the Xbox SmartGlass application, our console makes the phone the primary input device, and unlike Wii U's gamepad that only one player in the group can use, our system allows each player to have their own screen. This opens up for new and more innovative ways for the users to interact with the games. Furthermore, people carry their smartphones with them at all times, ensuring there will be enough controllers available to meet the demand.

2.2 Scope and contributions

Smartphones and computers come in many varieties, with different hardware and software configurations. In order to serve a broad audience it is important to build a general platform that can accommodate as many devices as possible. The aim of this project is to build a proof of concept platform. Thus, we only consider the Android operating system on the mobile side and the Microsoft Windows operating system on the computer. However, in the future we want the system to be platform independent.

When designing a platform used for gaming purposes, end user experience is of utmost importance. Given this, games must be designed with the intention of keeping a stable frame rate and controller input latency has to be kept at an acceptable level. Another important aspect is power consumption. If the system is designed without concern for power consumption, the smartphone's battery will be drained quickly. The user should not have to suffer from the frustrating experience of updating the application every time a minor change to a specific game is made. Therefore, this process of updating should be as automatic as possible. Given our focus on social interactions we will not use and code any single player games. Instead, we will focus on games which multiple persons can enjoy. These games will not have a set player limit and instead support any number of players.

This project has two main contributions. The design and development of the system (Chapter 3) and the implementation of the platform (Chapter 4). Each of these contributions will be addressed in the following chapters.

3 Design and Development

This chapter describes the design of the platform architecture, followed by an brief discussion of what libraries and languages are used, and why.

3.1 Designing a platform

The design decisions that are made when designing the platform are discussed in this section. It contains the requirements of the system as well as a motivation for using remote code execution (RCE) and how it is realized on the platform.

There are several hardware components in a smartphone which make it an interesting input device for games. The touch screen, accelerometer and camera all present different opportunities in terms of user interaction. Thus, a functional requirement of the system is to be able to read input from a number of hardware types, including the ones mentioned above and more. The system also needs a way to present contextual information on the touch screen.

In order to cater for different types of smart phones, with different operating systems, the design of the system needs to be created with platform-independence in mind. To better accommodate the needs of the intended audience (attendants of a social event); it is of utter importance that the process of setting up and starting a game is as quick and simple as possible. Given the requirements presented above, a basic system architecture can be created. A fundamental version of the mobile application needs to:

- Connect to the game console (in our case, the computer)
- Transmit input information
- Show graphical user interface

The first two items in the list concern network programming, which will be discussed in depth in section 4.1. However, the last point has direct implications for the fundamental design of the system. If the conventional way of building applications is followed, where a GUI has to be created separately for each mobile operating system, the system cannot be considered platform independent. To be able to provide the same graphical interface for more than one mobile operating system, a common platform must be built. But where does the graphical interface reside, if not on the mobile? A logical answer is the server, the only node in the network that all phones communicate with (see client-server model subsection 4.1.1).

3.2 Remote code execution

In this subsection we will discuss what remote code execution is and how the platform makes use of it.

The server is responsible for managing all game and controller logic, as well as the graphical interfaces for both the server and the smartphone controllers. The parts of the code that concern the smartphone controllers will (once a connection between the player's phone and the server has been established) be sent over the connection, received by the phone and executed directly afterwards. This process, where code is first sent from one computer to another, and then executed, is commonly referred to as remote code execution, or RCE in short [3]. It is a fairly well-known method which can and has been used to perform potentially illegal actions in poorly secured IT systems. There are of course legitimate, non-malicious, applications for RCE as well. A good example of this is online systems for learning new programming languages. The user in such a system is typically asked to enter some lines of code in the relevant language, which is then sent and executed on the server, yielding a result which is then sent back to the user. This makes it possible to get a hands-on experience with a language, without having to install a compiler, virtual machine or other additional software on the computer. An example of such a system is Try Clojure [4].

3.3 Client development

This section gives a summarized overview of the Android application. It mainly identifies which languages are used.

The Android client is written in Java, C/C++ and Lua. Lua is a dynamically typed language intended to be used as a scripting language [5]. Due to this, Lua is compact and easily embeddable. The specific Lua implementation chosen is LuaJIT, since it is arguably faster than all other Lua implementations [6] and runs on most platforms [7].The reason we decide to include C/C++ and Lua, and not simply write the entire client in Java, is that we want as much code as possible to be easily portable to the iOS operating system. C/C++ is natively supported by iOS and is supported by Android through the Android Native Development Kit (NDK) [7]. We use Lua to extend the client.

4 Platform

This chapter describes the core of the platform, starting with what is perhaps the most critical part - the network. It doesn't talk much about the Simple Data Language used for serialization, and the game related technologies concerning graphics and sound.

4.1 Network

One of the most, if not the most important part of the platform, is the network implementation. This section will discuss the various methods and problems associated with implementing a local client-server model, based on synchronous and asynchronous usage of the transmission control protocol (TCP).

4.1.1 Client-server model

A goal of the project is for the computer to emulate a video game console and the smartphones to emulate game controllers. The controllers of normal video game consoles do not communicate with each other. This applies to our system as well, and leads to a classic client-server network model. The computer (the console) acts as the server and the smartphones (the controllers) act as clients. Given this there is no direct communication between the phones. Any possible phone to phone communication has to go via the computer. In the network section, the word client refers to a smartphone connected to the local network and server refers to the computer in the local network which the clients connect to.

4.1.2 Bluetooth versus Wi-Fi

The computer and the phones need to be connected wirelessly in order for gameplay to be practical. There are two major competing standards for consumer grade wireless communication, Wi-Fi and Bluetooth. Both of them have their distinct advantages and disadvantages which we present below.

1. While Bluetooth is available in most laptops, it is not available in all desktop computers. Even if a desktop computer does not support Wi-Fi, it is probably connected to a router in the local wireless network via an Ethernet cable, and can thus be connected to. Therefore, Wi-Fi is more prevalent than Bluetooth.

2. Bluetooth uses different protocols than the ones given by the Internet Standard (the protocols in use on Wi-Fi). These protocols are less documented and harder to find tutorials for.

3. Bluetooth handles server discovery through the somewhat cumbersome Bluetooth pairing process. Such a process does not exist on Wi-Fi and has to be implemented by the programmer herself.

4. Bluetooth consumes less power [8].

After evaluating the properties of the two technologies, we came to the conclusion that Wi-Fi was a better fit for our purposes.

4.1.3 Connection

When a server has been found, a client can proceed to make a connection. The connection is made in a series of steps:

1. A client makes a TCP connection attempt to the server.

2. The server accepts the incoming connection.

3. The server gives the incoming connection pending status.
4. The server generates a unique session ID and associates it with the incoming connection.
5. The server sends the session ID to the client.
6. The client receives the session ID.

   At this stage is it possible that the connection is a reconnection. A client has lost connection and needs to reconnect. In the following list, items labeled by “a” specify a connection attempt, and the items labeled by “b” specifies a reconnection attempt.
7. a The client sends the received session ID to the server.
8. a The server receives the session ID.
9. a The server changes the connections status to active from pending.
10. a The connection process is done.
11. b The client discards the session ID.
12. b The client sends an old session ID to the server.
13. b The server receives the old session ID.
14. b The server checks if it remembers the old session ID from a previous connection.

    Depending on whether the server remembers the old session ID or not, the server needs to accept or reject the incoming connection. These different scenarios are described with the “c” (accept) and “d” (reject) subscripts.
15. c The server has used the old session ID, the server changes the connection status to active from pending.
16. c A message is sent from the server specifying that the reconnect was accepted.
17. c The reconnection process is done.
18. d The server has not used the old session ID.
19. d The server sends a message to the client that the reconnect was not accepted.
20. d The server forcefully closes the connection.

4.1.4 Disconnection

This subsection describes the different ways a connection might be dropped and what action is performed in each case.

Client side

Client will disconnect from the server under any of the following circumstances:

1. The Android activity enters an idle state.
2. An invalid message is received.
3. The phone receives a special shutdown message from the server.

4. The phone exits the game.
5. Timeout. A message has not arrived from the server for 30 seconds.

If a phone disconnects due to the activity entering an idle state (1), it will reconnect again when the activity once again enters an active state. If an invalid message is received (2) the phone preemptively exits the game. If the disconnect is caused by the shutdown message (3), the phone will exit the game. In the event of a normal exit (4) the phone simply closes the connection via standard TCP shutdown before the game exits. Lastly, if the server times out (5) what to do is left for the specific game to decide. This usually happens if a server was shut down incorrectly or if the phone got disconnected from the local network. In this case it is not clear how the client should react, which is therefore left to the specific game to decide.

Server side

A connection will disconnect from the server in the following circumstances:

1. The server is shutdown.
2. An invalid message is received.
3. Remote TCP shutdown request.
4. Timeout. A message has not arrived over the connection for 30 seconds

If the disconnection happens due to the server being shut down (2) the server will close the connections via a special shutdown message. If a message which does not conform to the message specification given in the next section is received by the server (2) it is assumed that the remote client is corrupt and the connection is terminated via TCP shutdown. The connection will be closed if the remote client requests connection termination via TCP shutdown (3). If a timeout occurs (4) the connection is closed via standard TCP shutdown.

5 Future work

This chapter concerns possible future work, which if pursued will make the platform even more appealing to consumers and developers.

5.1 More platforms

To make the system available to a wider audience, more platforms need to be supported. Most urgently, the application should be ported to the iOS operating system on the mobile side, and Mac OSX operating system on the server side.

5.2 User testing

In order to evaluate the functionality of the system as a whole and to evaluate the methods of controlling the created comparative, player testing needs to be carried out.

5.3 RCE sandboxing

The current system is not taking any security measures. The implications of this are that a malicious developer may execute arbitrary code on the connected smartphones, which of course cannot be tolerated. To consolidate this flaw, RCE sandboxing can be used.

6 Conclusion

The number of people who own smartphones is growing at a fast pace, and has been projected to reach 5.6 billion in 2019 [1]. Smartphones include many advanced hardware capabilities that can be utilized in a large number of ways. This project aims to make use of these devices to the fullest extent possible, by creating a virtual game console which uses smartphones as controllers.

There are also drawbacks with this approach. This is mainly due to the fact that smartphones lack many important hardware components present in modern day controllers. These include things like thumb sticks and hardware buttons. More avid players have come to expect these on a game controller making the transition awkward. However, there are many players who only play games on their smartphone and these players will have no problem using them on this platform.

References

[1] Ericsson. (2013, November) Ericsson Mobility Report. [Online]. https://www.ericsson.com/res/docs/2013/ericsson-mobility-report-november-2013.pdf

[2] Caroline J. Kistin, Barry Zuckerman, Katie Nitzberg, Jamie Gross, Margot Kaplan-Sanoff, Marilyn Augustyn and Michael Silverstein Jenny S. Radesky, "Patterns of Mobile Device Use by Caregivers and Children During Meals in Fast Food Restaurants," Pediatrics, Mar. 2014.

[3] Hannes Holm, Mathias Ekstedt Theodor Sommestad, "Estimates of success rates of remote arbitrary code execution attacks," Information Management Computer Security, vol. 20, no. 2, pp. 107-122, 2012.

[4] Anthony Grimes et al. (2014, May) tryclojure. [Online]. https://tryclj.com/

[5] Luiz Henrique de Figueiredo, Waldemar Celes Roberto Ierusalimschy. (2014, May) Lua 5.1 reference manual. [Online]. lua.org/manual/5.1/manual.html

[6] Mike Pall. (2014, May) LuaJIT performance: arm. [Online]. www.luajit.org/performance$_{a}rm.htmlMikePall.(2014, May)LuaJITcompatability.[Online].luajit.org/luajit.ht$

[7] Google. (2017, April) Android NDK. [Online]. https://developer.android.com/tools/sdk/ndk/index.html

[8] Yu-Wei Su, Chung-Chou Shen Jin-Shyan Lee, "A Comparative Study of Wireless Protocols: Bluetooth, UWB, ZigBee, and Wi-Fi," in Industrial Electronics Society, 2007. IECON 2007. 33rd Annual Conference of the IEEE, Taipei, 2007, pp. 46 - 51.