Replacing Cellular with WiFi Direct Communication for a Highly Interactive, High Bandwidth Multiplayer Game

by

Pablo Ortiz

B.S., University of California, Santa Barbara, 2011

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2013

Author .....
Department of Electrical Engineering and Computer Science
August 6, 2013

Certified by .....
Li-Shiuan Peh
Professor
Thesis Supervisor

Accepted by .....
Leslie A. Kolodziejski
Chair, Department Committee on Graduate Students

Replacing Cellular with WiFi Direct Communication for a Highly Interactive, High Bandwidth Multiplayer Game

by

Pablo Ortiz

Submitted to the Department of Electrical Engineering and Computer Science
on August 6, 2013, in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering and Computer Science

Abstract

The objective of this work is to explore the benefits of replacing cellular with Wi-Fi Direct communication in mobile applications. Cellular connections consume significant power on mobile devices and are too slow for many highly interactive mobile applications. Wi-Fi Direct, the recently released wireless standard, promises to provide the speed, power efficiency, and security of Wi-Fi to devices communicating within a short range. Using Wi-Fi performance as a baseline, the performance of a proof of concept system, Super Tux Kart Direct, is evaluated when communication is enabled by either LTE or Wi-Fi Direct. At the time of writing and to the best of the author's knowledge, Super Tux Kart Direct is the first, real-time, multiplayer kart racing game playable via Wi-Fi Direct.

Thesis Supervisor: Li-Shiuan Peh
Title: Professor

Acknowledgments

The completion of this thesis would not have been possible without the tireless support and encouragement from my dear Katie Lampier. My sanity and morale were brought back from the brink countless times by her efforts.

Without the expertise of my colleague Jason Gao, I would have floundered under the weight of my ignorance with respect to many of the practical considerations involved in conducting research on Android devices. His aid was invaluable in ensuring that this work progressed smoothly.

I would be remiss if I failed to mention the noble efforts of Dr. Seth Hetu in assisting me with the completion of this thesis. Not only did he spend many hours of his valuable time helping me test and debug my proof of concept system, but he also gave me a great deal of practical advice for dealing with large code bases.

I would also like to extend my deepest gratitude to my thesis supervisor, Dr. Li-Shiuan Peh, for all the help she provided me with while working on this project. Her wisdom and guidance got me through many of this work's major roadblocks.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Humanity, as a whole, is more connected now than ever before, and, as time goes on, becomes more connected by all forms of electronic communication. As of 2013, the total number of cellular subscriptions in the world is nearly equal to the global population [9]. This near-saturation of cellular connections around the globe coupled with high consumer demand highlights the need for fast and efficient communication methods in mobile devices. While service provided by 4G cellular networks is sufficient for many applications, its shortcomings inhibit the deployment of many others.

4G cellular networks provide cellular devices with higher data rates and lower latencies than their 3G predecessors. The improved network performance comes with a high cost for any cellular device communicating over a 4G network, however. A recent study has shown that, in the case of LTE, this improvement in network performance takes its toll on the batteries of cellular devices, reportedly consuming as much as 23 times as much power as communicating over Wi-Fi. [8] The same study showed that connections to LTE networks also consume more power than connections to 3G networks. Though 4G networks sport better network performance than their 3G predecessors, 4G networks still do not provide data rates as high as those observed in Wi-Fi connections, one of the most power efficient and speedy communication methods available to users. Furthermore, the impending spectrum crunch along with exponential growth in demand suggests that the cellular infrastructure may face intense scalability challenges. Fortunately for applications in which cellular device locality can be leveraged, the new, short range, wireless technology Wi-Fi Direct could be used to mitigate the shortcomings of communication over cellular networks.

Wi-Fi Direct builds on Wi-Fi and inherits the data rates, security features, and power efficiency of its predecessor. [5] The standard works by allowing Wi-Fi enabled devices to connect to a Wi-Fi Direct enabled device as though it were a wireless access point. Devices connected in this fashion are then able to communicate directly in a peer-to-peer fashion as if a part of a network with a star topology. Wi-Fi Direct makes use of the same frequency bands as Wi-Fi does. Thus, use of the technology will not add interference to nearby devices communicating over a cellular network. As an infrastructureless technology, Wi-Fi Direct is able to enable communication in places where existing infrastructure does not provide coverage (i.e. remote areas, disaster zones, underground, etc.). It also has the potential for reducing the load of cellular networks by removing the need for the cellular network as an intermediary in applications where device locality can be leveraged.

The objective of this thesis is to explore the benefits of using the new, short range, wireless technology Wi-Fi Direct as a means of communicating between cellular devices instead of traditional communication over a cellular network. The design and implementation of a proof of concept system is presented here along with its performance results when either a 4G connection or a Wi-Fi Direct connection is used.

1.1 Thesis Organization

The rest of this thesis will be organized as follows.

Chapter 2 provides context for the communication technologies considered in this work. It presents the history, strengths and weaknesses of many common wireless technologies used in cellular devices. Perhaps most importantly, it provides many more details on the features and performance of Wi-Fi Direct.

Chapter 3 presents the design of the proof of concept system, Super Tux Kart Direct, and key aspects of its implementation.

Chapter 4 describes the equipment used during development, the equipment used during experiments, and the experimental setup. The description of the experiments conducted in this work and presented in this chapter is necessary background reading in order to make sense of the results presented in the chapter that follows it.

Chapter 5 presents the results of all the experiments.

Chapter 6 presents the conclusions drawn from the results and possible implications. It

also contains within it a discussion of possible future work.

Chapter 2

Background

Cellular phones have only been commercially available since the introduction of the DynaTAC 8000x into the market in 1983. In these past 30 years, these devices have evolved from being little more than modified, hand-held radios to powerful, mobile computers. The potency of the cellular phone as a communication device has seen so much improvement that what was once considered an impractical luxury is now often considered a daily necessity.

A modern cellular phone is often capable of allowing its user to communicate via a wide range of technologies and to connect to a wide range of devices. Despite this fact, the best communication technology for a particular type of application is not always considered when designing a new mobile application. What follows is an overview of the history, strengths, and weaknesses of the prominent communication technologies available on current mobile phones.

2.1 Wi-Fi

The Wi-Fi Alliance defines Wi-Fi as a technology that grants wireless network connectivity to devices enabled by the various Institute of Electrical and Electronics Engineers (IEEE) 802.11 radio technologies [4]. The original form of the IEEE 802.11 standard was released in 1997 and revised in 1999. Over the years, the original 802.11 standard has been further revised and has spawned a whole family of wireless standards (802.11a, 802.11b, 802.11g, 802.11n, 802.11ac[1], and 802.11ad). The most common use for Wi-Fi in cellular phones

[1]This standard is scheduled to be approved and published by early 2014. [1]

today is to enable connections to wireless access points that grant access to the Internet. Whenever the term Wi-Fi is used in the remainder of this thesis, it will be referring to this specific use of the technology.

Wi-Fi is often the most optimal choice for a mostly stationary cellular phone user who wishes to communicate in some fashion via their cellular phone. Assuming that a user is near to and has access to a wireless access point connected to a high-speed internet connection, Wi-Fi can provide a cellular phone user with a low latency, high throughput communication experience that consumes less power than cellular connections. It is a reliable technology with much in the works in terms of future improvements.

Unfortunately, the conditions under which Wi-Fi can be effectively used make it difficult to take full advantage of the great network performance and power efficiency the technology offers. For instance, a Wi-Fi connection can only provide network performance comparable to the kind of Internet connection the wireless access point the Wi-Fi connection is derived from has. To even obtain a Wi-Fi connection, a cellular phone user needs access to some nearby wireless access point. This can be difficult because, while there are some businesses, schools, and even entire cities that offer free access to wireless access points on their premises, most wireless access points are private and password protected. Furthermore, the range of a typical access point is quite short, so a cellular phone user cannot travel very far from an access point they are connected to without dropping their connection. Taken together, these requirements severely limit the availability of a Wi-Fi connection for cellular phone users on the go. Fortunately, a traveller with a cellular phone has other communication technologies available to them on their device that enable communication in more places than Wi-Fi.

2.2 Cellular Networks

The technologies that enable communication over cellular networks have changed considerably since the cellular phone was first introduced to the consumer market. For instance, cellular connections can now grant users both regular telephone service and Internet connectivity as needed. Many mobile communication technologies and the standards that defined them have come and gone while others have remained and evolved to match consumer demand. Throughout all this change in the technological landscape, the quality of cellular

communication overall has undergone steady improvement for quite some time.

A new generation of mobile communication technology has appeared, roughly, every decade since cellular phones were made commercially available. Each generation has brought new features and better network performance in terms of bandwidth, throughput, availability, etc. to the consumer. That said, the exact features that distinguish one generation's technology from another is somewhat ambiguous. There is no universally accepted set of guidelines for precisely classifying one technology as being part of a particular generation over another. The ITU-R's labels for existing mobile communication technologies have garnered wide consensus from both industry and consumers, however. The ITU-R and others have been releasing specifications for new technologies and the standards to classify technologies into new generations for many years. So far, the mobile phone industry has successfully met many of the goals laid out by these specifications and standards.

Commercially deployed cellular networks have already met or exceeded the requirements set out by the widely agreed upon 1G, 2G, and 3G standards. They have not done so, however, with respect to the standard released for the fourth generation of cellular communication technology (4G) released by ITU-R in 2008. The networks that have deployed the first-release versions of Mobile WiMAX and LTE are the closest to offering a next-generation mobile experience. Though branded otherwise, they are not true 4G technologies but are, instead, the forerunners of future 4G technologies. For the sake of consistency with respect to the vernacular of mobile technology, Mobile WiMAX and LTE will be referred to as 4G technologies for the remainder of this thesis.

2.2.1 Mobile WiMax

The Mobile WiMAX standard has its origins in the release of the IEEE 802.16e-2005 standard which itself is an extension of what is now called Fixed WiMAX. It was originally designed to provide 30-40 Mbps data rates to consumers on the go. However, as with all wireless networks, the actual data rates experienced by a Mobile WiMAX user depend greatly on environmental factors. Implementations of the standard have been deployed in many countries worldwide, and the standard itself is a direct competitor to the LTE standard for global dominance.

Unfortunately, the performance of Mobile WiMAX networks will not be considered in this thesis. The phones used to run experiments in this work were only fit to connect to

LTE networks. Mobile WiMAX was mentioned here for completeness.

2.2.2 LTE

The Long Term Evolution (LTE) standard is the pre-4G standard developed by the 3GPP that evolved from the GSM/EDGE and UMTS/HSPA network technologies. The new standard offers greater network capacity and data rates over its predecessors by leveraging new digital signals processing techniques and a redesigned network architecture. The standard was first proposed in 2004 by NTT DoCoMo and was finalized in 2008. In 2009, the first publicly available LTE service was deployed, and, ever since then, the number of deployed LTE services has increased significantly across the globe.

Recently, there has been legitimate concern over the dwindling amount of wireless spectrum available to cellular providers for offering their services to consumers. The LTE standard attempts to mitigate this problem by making more efficient use of available spectrum. Access to an LTE network is achieved using OFDMA (a multi-user version of OFDM) for the downlink and SC-FDMA for the uplink. These multiplexing techniques are purportedly more spectrum efficient than those used in 3G networks and, in the case of SC-FDMA, power efficient. While improving the spectrum efficiency of cellular networks does increase their capacity for providing service to consumers, it is a temporary fix for a looming problem. The industry must find other solutions to meet escalating consumer demand.

The splendid network performance that can be observed in real world LTE deployments is one of the LTE standard's greatest strengths. The downlink and uplink speeds on LTE networks have truly begun to rival those observed when a device is connected to a WLAN [8]. As LTE is most efficient when a lot of data is being sent over a connection for a significant period of time, LTE connections are provably great for audio/video streaming services such as Spotify or YouTube.

The LTE standard is not without flaws, however. For instance, it, like all cellular network standards, has a problem with availability of service, though it is minor compared to Wi-Fi's problem with availability. It is only possible to connect to an LTE network when there are cell towers nearby. Connections to LTE networks have also been found to be quite power inefficient. While it is almost to be expected that LTE connections would be less power efficient than Wi-Fi connections, they have also been found to be less power efficient than their 3G predecessors. [8] Furthermore, an LTE connection does not achieve

optimal power efficiency levels until large data packets are being sent consistently over a connection that has been open for an extended period of time. It is quite inefficient when only small packets are being sent across a connection that is open for only a short period of time. This implies that an application like Twitter that needs to connect to a source over a network frequently to retrieve small bits of information may make suboptimal use of a phone's resources.

2.3 Device-to-Device Technologies

4G networks and Wi-Fi share some similar problems. Both require the deployment of infrastructure to provide fast, reliable service over as wide an area as possible. Though availability of service is not usually a problem in areas with high density populations, areas with more sparse populations might have little to no communication infrastructure and, thus, no service. Furthermore, communication over a cellular network or routed through a Wi-Fi access point may introduce unnecessary latencies when two or more cellular phones are trying to communicate within close proximity of one another. Technologies that enable device-to-device communication could mitigate these problems.

Device-to-device communication does not require the deployment of any costly infrastructure to function. As long as the devices communicating are equipped with the requisite hardware and software, communication can proceed. This means that device-to-device communication can occur just about anywhere. Furthermore, device-to-device technologies are often quite power efficient in terms of the amount of power consumed on individual devices when compared to power consumed by cellular connections. If the power consumed by cellular network infrastructure or the nodes used to route packets through the Internet are considered, then device-to-device communication is probably much more power efficient than both 4G and Wi-Fi. What follows is an overview of the history, strengths, and weaknesses of the some of the most relevant device-to-device technologies available on current mobile phones. One of these technologies is the subject of this thesis.

2.3.1 Ad-hoc Wi-Fi

An additional mode of operation is defined in the IEEE 802.11 standards in addition to Wi-Fi's normal (infrastructure) mode of operation. This mode is called ad hoc mode, and

it allows Wi-Fi enabled devices to communicate directly without having to go through a wireless access point. Instead, a device in ad hoc mode sends, receives, and routes data in a dynamic peer-to-peer network of devices that have all been configured to communicate in ad hoc mode with the same channel number, SSID, etc. As ad hoc mode is a standard part of the 802.11 standards, the ad hoc mode of operation was available on mobile devices at precisely the same time that Wi-Fi in infrastructure mode became available. The use of this technology in mobile devices spawned a great deal of research in the mid-90s on what is called Mobile Ad Hoc Networks (MANETS). It was likely the many nice features ad hoc mode offers mobile devices that inspired the swell in research.

In some respects, Wi-Fi in ad hoc mode is comparable to Wi-Fi in infrastructure mode. In terms of power efficiency, communication over an ad hoc network formed by devices with ad hoc mode enabled is superior to communication over LTE and 3G networks just like in infrastructure mode. Depending on the specific device and environmental conditions, the effective communication range of Wi-Fi in ad hoc mode and Wi-Fi in infrastructure mode can also be quite comparable. That is where the similarities end, however.

There are a number of characteristics of Wi-Fi in ad hoc mode that make it an undesirable technology to work with on mobile devices. For example, ad hoc mode cannot be enabled at the same time as infrastructure mode. In most cases, this is a large problem as Internet access would only be possible via a cellular connection. Also, the bandwidth and the throughput of Wi-Fi in ad hoc mode are inferior to Wi-Fi in infrastructure mode. Past a certain point, network throughput significantly decreases below that of infrastructure mode if more devices join an ad hoc network or if devices in an ad hoc network move farther apart. The biggest problem with using Wi-Fi in ad hoc mode on cellular phones, however, is that the mode itself is made inaccessible by mobile operating systems. [10] To enable Wi-Fi in ad hoc mode on cellular phones, the phone must be rooted and have a software patch installed. It is only then that Wi-Fi in ad hoc mode can be enabled in code. As both consumers and the industry itself are not very interested in the technology, this state of affairs is not likely to change in the future.

2.3.2 Bluetooth

The Bluetooth standard was originally designed as a wireless alternative to RS-232 communication. It is a popular technology used in such devices as wireless headsets for cellular

phones and video game controllers for current generation consoles. Connections between Bluetooth devices follow a master-slave structure in which one device is the master of up to seven slave devices. These personal area networks are formed when one Bluetooth enabled device tries to connect to another Bluetooth enabled device via a process called pairing. Among other things, the pairing objects negotiate which device is the master and which is the slave at the time of the pairing. Once pairing is complete, the two devices can communicate reliably and securely over short distances. It was created in 1994 at Ericsson and has since undergone many upgrades and revisions. The most recent version of Bluetooth, version 4.0, includes Classic Bluetooth, Bluetooth High Speed, and Bluetooth Low Energy protocols. These features, as well as others, make Bluetooth an attractive tool for use in application development.

The performance of Bluetooth on a particular device depends on a number of things. The Bluetooth power class of a device defines how much power is consumed when the device is communicating over Bluetooth and the range at which the device can communicate to other Bluetooth enabled devices. There are three power classes. Class 1 devices have the longest range (100 m) and consume the most power (100 mW). Class 3 devices offer the other extreme. They have the shortest range (1 m) and consume the least amount of power (1 mW). The vast majority of Bluetooth enabled devices are of power class 2, a compromise between the two extremes. Class 2 devices have a range of 10 meters and consume 2.5 mW. Most mobile phones are class 2 devices. [15] Both class 2 and class 3 Bluetooth devices consume considerably less power than Wi-Fi. If a device offers an implementation of Bluetooth with the high speed protocol enabled, Bluetooth can theoretically operate at data rates as high as 24 Mbps. Thus far, Bluetooth, among other things, has found great success as a technology for enabling the connection wireless peripherals to laptops, cellular phones, etc.

The caveats that come with using Bluetooth could inhibit its use in other applications, however. The Bluetoth Low Energy protocol, for example, is designed to work well with the power constraints of coin cells while still providing the range of the Classic Bluetooth protocol. It is able to operate under such constraints by allowing for low duty cycles and targeting specific applications. The Bluetooth High Speed protocol allows for theoretical data rates of up to 24 Mbps. This is a little misleading, however, as the Classic Bluetooth is used to set up the connection, and, then, the actual data transactions are carried out

over an 802.11 link. Not all devices support the High Speed protocol, unfortunately. Those that do not can only achieve a maximum theoretical data rate of 3 Mbps which is much less than what could be obtained with a cellular or Wi-Fi connection. Despite these caveats, Bluetooth is a solid technology to use for applications that must or are designed to sometimes communicate directly with other devices.

2.3.3 Wi-Fi Direct

Wi-Fi Direct is a relatively new wireless standard released by the Wi-Fi Alliance. It allows devices to connect to each other directly at Wi-Fi speeds without the need for a wireless access point. Unlike with Wi-Fi in infrastructure mode, the technical minutiae of establishing a connection is largely obscured to the user of a Wi-Fi Direct device. To begin the process of forming a connection, the Wi-Fi enabled device initiating the connection typically needs to simply select a connect button, and the Wi-Fi Direct enabled device that is being connected to simply needs to either select an acknowledgement button or enter a PIN. It is important to note that only one device needs to be Wi-Fi Direct enabled for communication to be possible over Wi-Fi Direct. All devices must be Wi-Fi enabled, however. After the connection request has been sent, a Wi-Fi enabled device that is trying to connect to a Wi-Fi Direct enabled device must negotiate which of the two devices will act as the group owner. The device that then takes on the role of the group owner acts as an access point for other devices to connect to. In the case where the connecting device is not also Wi-Fi Direct enabled, the device that is Wi-Fi Direct enabled would likely be elected group owner by default. In this way, relatively power efficient, high throughput networks can be formed.

As recently as 2010, the Wi-Fi Alliance was already certifying products as being Wi-Fi Direct certified. At this current point in time, the adoption rate of the Wi-Fi Direct standard only seems to be increasing. Many televisions and cellular phones have adopted the technology, and even the next generation video game console, the Xbox One, has Wi-Fi Direct support listen in its technical specifications. Few applications have been developed with Wi-Fi Direct in mind despite widespread industry support and the technology's host of good features.

The Wi-Fi Direct standard is very powerful and versatile. As it is built on Wi-Fi, it has inherited some of its characteristics. Wi-Fi and Wi-Fi Direct are roughly the same in terms of power efficiency, though Wi-Fi Direct may do better than Wi-Fi depending on the exact

802.11 protocol being used. Optionally, Wi-Fi Direct enabled devices can maintain both a Wi-Fi Direct connection and Wi-Fi connection. This optional feature could allow for such use cases as a single cellular phone providing internet access to a network of cellular phones nearby to conserve power in the non-group owner devices. Furthermore, like Wi-Fi, it also uses the WPA2 security protocol to protect communication between devices. The Wi-Fi Protected Setup (WPS) feature is also employed as a security measure. In terms of network performance, the Wi-Fi Direct standard is supposed to offer connection speeds of up to 250 Mbps. The standard also purports to offer a range of up to 200 yards. Wi-Fi Direct has few disadvantages as a device-to-device wireless technology, but there remains significant concerns that could deter widespread adoption.

For instance, The WPS standard used to simplify the establishment of a Wi-Fi Direct connection was revealed to have a major security flaw in 2011. This flaw allows an attacker to bypass WPA2 protections within hours by remotely obtaining the WPS pin and, by extension, the WPA/WPA2 pre-shared key used by Wi-Fi Direct. Currently, the only available solution to this issue is to disable WPS. As for the developer adoption issue, Wi-Fi Direct has not been used in many applications beyond a few file transfer applications. Perhaps the industry as a whole is waiting to see what will come of the technology. However, to the best of the author's knowledge, the proof of concept system presented in this work is the first real-time, multiplayer kart racing game that makes use of Wi-Fi Direct.

Chapter 3

Design

To properly explore the benefits of using Wi-Fi Direct connections in applications as opposed to more common connection types, a proof of concept system was designed, implemented, and tested. It was decided that the proof of concept system would be a real-time, multiplayer game for a number of reasons. First, games are one of the most common and popular types of applications in all mobile app stores. Thus, a technology that proves useful in terms of game performance would likely effect swift and widespread change if adopted. Second, real-time multiplayer games often involve the constant, swift exchange of data amongst multiple entities to advance game state. In such a system, it should be possible to tweak a variety of in-game parameters for the purpose of properly gauging how a particular communication technology performs under different network conditions.

Specifically, the proof of concept system is a heavily modified version of Super Tux Kart that has been ported to Android. The most notable modification to the game was the addition of multiplayer modes of play that allow players to compete against each other over Wi-Fi or Wi-Fi Direct were added. The characteristics of the resulting networked, real-time, multiplayer kart racing game are the reason why Super Tux Kart modified in this way was chosen as the proof of concept system. As Super Tux Kart is a racing kart game, the game is required to carry out certain tasks to be considered to be executing well. For instance, the game must maintain the illusion of high speed movement to all players in a multiplayer session. To do this, the game state on each player's device must appear to advance swiftly, smoothly, and without interruption. Updates to the state of each player must propagate through the network to all players very quickly to ensure this

speedy advancement of overall game state. Assuming that a particular racing kart game responded quickly to incoming data, the illusion of speed on all player's screens would only be dependent on the latencies for transmitting data over whatever type of network is being used. A racing kart game must also portray a consistent view of the game world to each player no matter how many objects are flying around a race track. Typically, in a racing kart game, players can acquire power ups that shoot projectiles to hinder the progress of other players. A similar requirement to maintain the illusion of speed applies to these projectiles as well. However, another problem presents itself with respect to these kinds of power ups. An arbitrary number of these projectiles could be flying around the game world at any particular time. A particularly wacky racing kart game might offer power ups that allow players to launch projectiles ad infinitum. Sending all this state over a connection now becomes both a latency and bandwidth problem. The ability to stress Wi-Fi, Wi-Fi Direct, and LTE networks in this way made Super Tux Kart an attractive choice for tweaking into a proof of concept system. What follows is a discussion of the details of the system's design as well as notable implementation techniques for developers interested in building a similar system.

3.1 Super Tux Kart

Super Tux Kart is an open source computer game similar to Mario Kart written in C++ for the Linux, Mac OS, and Windows platforms. It is a kart racing game that offers many features that are now staples of the genre. For example, players can engage in a single-player story mode where they race against AI[1]-controlled karts at varying levels of difficulty in race tracks that get unlocked as the player accumulates points. They can also partake in a mode of play where they race in one-time single races or grand prixs against a mix of AI-controlled karts and any additional players whose input devices are all connected to the same computer. The race tracks themselves are littered with power-ups, and every kart, whether it be controlled by a human or AI, is driven by one of many iconic mascots of various open source projects. Over the long history of this game's development, these and many other features have been added to each successive release. As of 2013, Super Tux Kart remains in active development.

[1]Artificial Intelligence

3.1.1 History

Super Tux Kart has its roots in the open source project Tux Kart. Due to disagreements amongst the development team, the Tux Kart project collapsed. It was resurrected in its current form by Joerg Henrichs in 2006 [2]. Super Tux Kart was unplayable for some years after the project began. However, in its current state it is a fully fleshed out kart racing game with a wealth of features.

3.1.2 The Xapantu Version

The version of Super Tux Kart used to create the proof of concept system was not an official release version of the game. A Super Tux Kart developer, Xapantu, had already begun porting the game to the Android platform and had made their incomplete port available for download to the public. [3] It was this version of the game that was used as a base for the proof of concept system. For the remainder of this thesis, the base version of Super Tux Kart used to make the final proof of concept system will be referred to as the Xapantu version, and the single mode of play offered by this version of Super Tux Kart will be referred to as Xapantu mode.

The Xapantu version of Super Tux Kart is a bare-bones port of Super Tux Kart to Android. Like the proper release versions of the game, the Xapantu version starts with a loading screen that populates the lower part of the screen with icons to indicate the extent to which initialization has progressed. Unlike the proper release versions of the game, the Xapantu version immediately forces the player to engage in a race between 3 AI-controlled karts on the race track called Amazonian Journey once the initialization process has completed. There is no ability to pause the game, no menu, and, thus, no means for a player to change the kart, race track, or mode of play. This arrangement can be attributed in part to the fact that the Xapantu version of the game is still in the early stages of development and a single race against multiple AI-controlled karts is good enough to test many of the game's basic features. The fact that the vast majority of Super Tux Kart's in-game assets have not been refactored to optimize for a cellular phone's screen size and storage restrictions is likely another contributing factor. The subset of Super Tux Kart's assets that Xapantu chose to refactor are only those assets needed to fully render the Amazonian Journey race track, the Tux kart, the GNU kart, and the graphical effects

of all the various power ups. Once the race is over, the game exits.

Implementation

For Super Tux Kart to be successfully ported to Android, it was necessary for all of its non-STL[2] external libraries to be ported to Android as well. While porting the Bullet and ENet libraries to Android is straightforward [3], porting the Irrlicht graphics engine is not. The Irrlicht engine, out of necessity, has specific code for each operating system it supports. Neither Android nor any mobile operating system is supported by Irrlicht as of this writing. Thus, all available ports of Irrlicht for Android are unofficial and tend to have poor documentation, if any. Thus, it was amazing to have complete, working ports for the Bullet, ENet, and Irrlicht libraries packaged with the Xapantu version of Super Tux Kart.

Porting Super Tux Kart to Android was made much simpler with the recent addition of the NativeActivity class to the growing host of tools available to Android Native Development Kit (NDK) developers. The NativeActivity class, a subclass of the Activity class, delegates the implementation all its functionality to developer-defined C code in a well defined way through Android-provided C libraries. When developing an Android application in Java as normal, an Activity class is used both to provide a potential entry point for starting the application and to define the full functionality of one of the application's screens. Additionally, a developer can get complete access to all of a phone's sensors from an Activity class. [4] Developers were previously unable to make use of any of this functionality easily in native C code. This meant that they had to employ many more coding tricks to complete their application if they wanted or needed it to be primarily written in C. In the case of Super Tux Kart, direct access to graphical hardware resources and to some form of user input are needed in the standard version of the game. With the use of the NativeActivity class, it was possible to grant access to these resources to the game code without having to rewrite the game completely in Java or expend a great deal of effort performing coding tricks.

Some coding tricks were employed by Xapantu to further reduce the amount of refactoring required to port Super Tux Kart to Android. The one trick that did this best was

[2]Standard Template Library

[3]One only needs to rewrite the build rules in the make files for these libraries into build rules that comply with the format of Android.mk files.

[4]There are some other uses for the Activity class, but these are the main ones.

compiling Super Tux Kart as a shared object library. By adding a few functions to the code base and making them visible to code outside the shared object library, the entirety of Super Tux Kart could be executed more or less as-is by calling functions in the Super Tux Kart library from a NativeActivity callback function.

By opting not to rewrite Super Tux Kart from the ground up in Java and instead making use of the NativeActivity class to facilitate the port's development, Xapantu was required to significantly alter the way Super Tux Kart is executed. In the standard version of Super Tux Kart, the game progresses in a loop that only ceases execution when the player exits the game. In the Xapantu version of Super Tux Kart, the game could not be executed in this way due to the constraints of the Android Activity life cycle. An Android activity gives a developer the option to define the behavior of certain well-defined callbacks that are called by an untouchable UI thread. This UI thread will produce an error if any of these callbacks block the thread by never returning. Thus, Super Tux Kart cannot be executed in a NativeActivity callback by calling a single function from the shared object library that never returns, and running the game in its own thread would create a whole slew of synchronization problems. To resolve this issue, Xapantu changed the flow of execution of Super Tux Kart in the following way. After the initialization process, Super Tux Kart immediately returns before entering what would have been the original game loop. Instead, a function from the Super Tux Kart shared object library is called in a NativeActivity callback that advances Super Tux Kart by one iteration of the original game loop for every iteration of Android's UI loop. In this way, the Xapantu version of Super Tux Kart is able to execute properly on most Android devices[5].

3.2 Super Tux Kart Direct

The proof of concept system, Super Tux Kart (STK) Direct, builds off of the Xapantu version of Super Tux Kart. The overall design of the final version of STK Direct is shown below in Figure 3-1. At a very high level, it is a straightforward client-server design. Each client (phone) executes a complete version of the game and has a subset of its internal state authoritatively dictated by a server. In terms of Figure 3-1, the yellow box labeled Super

[5]If the Android device that the Xapantu version of Super Tux Kart is running on has a GPU that does not support the hardware generation of mipmaps, the vast majority of in-game textures will not load, and the in-game models will be some shade of grey

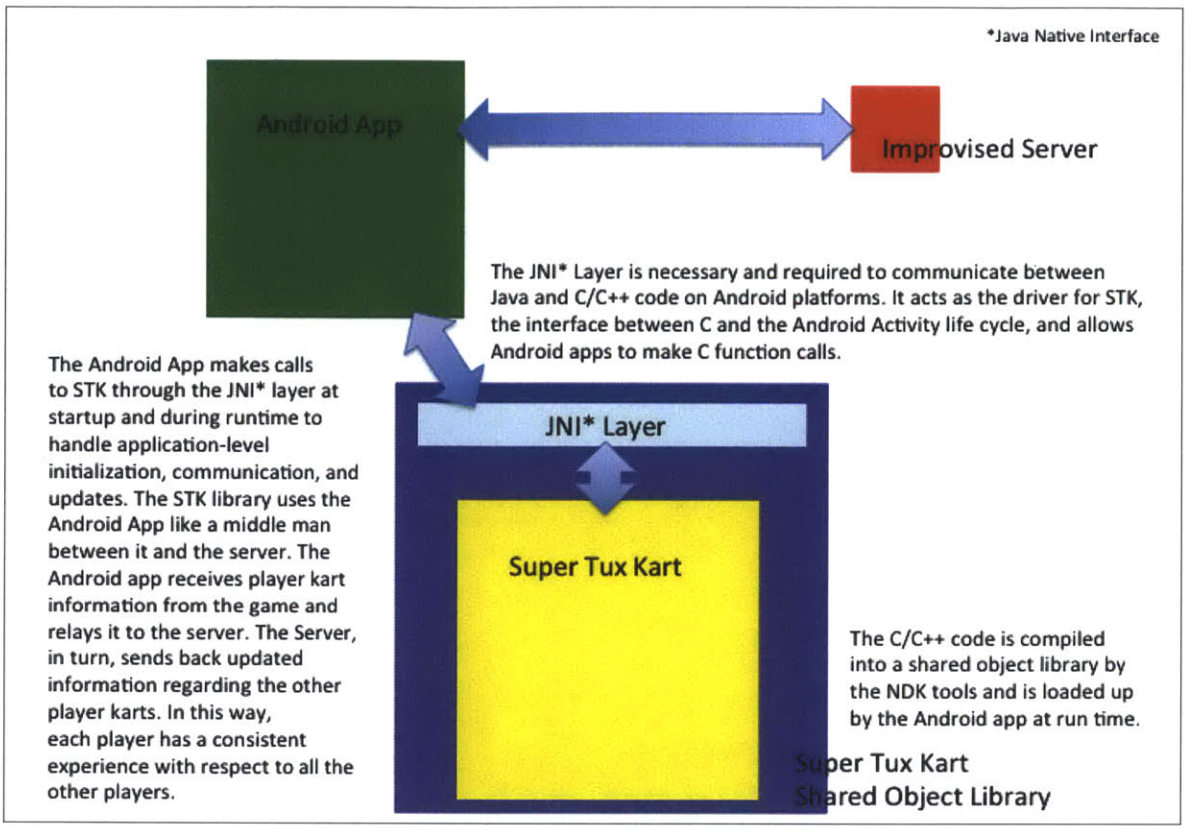


Figure 3-1: Design of Super Tux Kart Direct

Tux Kart is essentially the entirety of the Xapantu version of Super Tux Kart. Graphically, it is easy to see how significantly the Xapantu version of Super Tux Kart had to be changed to achieve the desired results. The design of the Xapantu version of Super Tux Kart was limited in terms of how much it could be improved by the choice to only develop the port in C. There is a wealth of features offered by Android, such as Wi-Fi Direct, that cannot be accessed from C on Android systems at this point in time. Thus, to add multiplayer functionality to the Xapantu version of Super Tux Kart, many changes and additions to the system had to be made.

The game itself was altered to have three modes of play. One of the modes of play is a single-player Super Tux Kart experience while the other two are multiplayer Super Tux Kart experiences. The available game modes are Xapantu Mode, Server Mode, and Wi-Fi Direct Mode. Xapantu Mode is the single-player mode of play available in the Xapantu version of Super Tux Kart. It was left in the final version of STK Direct for testing purposes. Server Mode is the mode of play where two or more players compete against each other in a race on the race track Amazonian Jungle over Wi-Fi. Specifically, the Wi-Fi connection on

each player's Android device is used to connect to a simple, remote game server manages the multiplayer game session. Similar to Server Mode, Wi-Fi Direct mode is the mode of play where two or more players compete against each other in a race on the race track Amazonian Jungle over Wi-Fi Direct. In Wi-Fi Direct mode, all the player's Android devices are connected to the device that is acting as a group owner. This group owner device acts both as another player in the Super Tux Kart multiplayer session and the game server. The multiplayer modes of STK Direct, Server Mode and Wi-Fi Direct Mode, are the primary focus of the system and the means by which Wi-Fi, LTE, and Wi-Fi Direct will be compared to one another.

To enable multiplayer modes of play in Super Tux Kart on Android using the Xapantu version as the base, it was necessary to add a few layers of software on top of the original game. Android-specific Java code was added on top of the C code so that the game would be able to make socket connections, access Wi-Fi Direct, let a user set execution preferences, and let a user control when the actual game launches. By itself, this is not all that useful as Java code and C code cannot communicate directly without assistance. With the use of the Java Native Interface (JNI), however, it becomes possible for data to be exchanged between Java code and C code. The JNI allows Java methods with special declarations to be implemented as C functions. By using a host of JNI functions and NativeActivity callbacks implemented in C as intermediaries, data is passed back and forth between the Android-specific Java code and the Super Tux Kart shared object library. This chain of communication is essential for enabling multiplayer play. Data from the Super Tux Kart shared object library can only be sent to the game server from Java code, and all of the game server's replies can only be sent back to the Super Tux Kart shared object library through the Java code.

3.2.1 Wi-Fi Server

The Wi-Fi server or the baseline server is the server used to manage multiplayer Super Tux Kart sessions in Server Mode. It runs on a remote machine accessible via a Wi-Fi or LTE connection. The server was not designed to be used by a great number of users so it only handles one game session at a time. The major duty of the Wi-Fi game server is to ensure that the game state of one client is consistent with the game state of all the other clients at every time step.

To maintain consistent game state across all clients, the game server executes in lock step. A simple barrier is used to prevent progress until certain conditions are met on the server side, and each client blocks until it receives a response from the server. For a new multiplayer game session to begin, each client device must first connect to the server. Once all clients have connected to the game server, the game server signals that the game can commence. At this point, each client device is required to submit the state data of the kart that is being directly controlled by the player using that device. Once state data is received from each client device, all of the received data is aggregated into and sent back to each client device. A client that receives this response from the server is able to advance the Super Tux Kart game loop by one iteration. The data provided by the server is the authoritative state of the karts controlled by other clients for the next iteration of the game. Each iteration of a multiplayer game of Super Tux Kart advances as described above.

The Wi-Fi server has a rudimentary ability to handle client disconnections in the middle of a session. If, for whatever reason, a client device disconnects from the server after a multiplayer Super Tux Kart game session has being, that client device is treated as though it has submitted its state data for all subsequent iterations of the session. The state transmitted for the disconnected client device to all the other client devices is precisely the same state as the disconnected device's kart had at the time of the disconnection. Effectively, all clients that are disconnected from a multiplayer game session are treated as if their karts were completely stationary.

3.2.2 Wi-Fi Direct Server

The Wi-Fi Direct server is the server used to manage multiplayer Super Tux Kart sessions in Wi-Fi Direct Mode. It works exactly the same way as the Wi-Fi server with a few minor differences. The first difference is that the server code runs directly on the Android device that is acting as the group owner for the Wi-Fi Direct group assembled to play a multiplayer Super Tux Kart game. The second difference is the way in which the group owner device executes its second role as a Super Tux Kart client. To make optimal use of resources, the group owner device does not open a socket connection to the server code as in the case of the Baseline Server. Instead the client Java code communicates with the server directly using public class methods.

3.3 Key Implementation Details

The most important technique used when implementing the STK Direct is also the one that, coupled with JNI, allowed C code to obtain indirect access to the full suite of features made available to Android developers in Java. The technique in question is subclassing the NativeActivity Java class. By subclassing NativeActivity in Java, a developer obtains all the benefits of a NativeActivity (being able to define Activity life-cycle functionality in C) as well as all the benefits of a standard Activity class. In STK Direct, for instance, this technique allowed the Java code to handle all the networking while the C code acted as the driver and interface to the Super Tux Kart shared object library. The JNI allowed the networking code and the driver code to coordinate their execution as necessary. To the author's knowledge, this is the only way to grant network access to a game written in C running on an Android device.

3.4 Related Work

To the best of the author's knowledge, this work presents the first, real-time, multiplayer kart racing game enabled by Wi-Fi Direct. Other research endeavors exist that evaluate or use Wi-Fi Direct in some capacity, however. Descriptions of some of these endeavors are briefly presented here. The studies [16] and [12] offer novel methods to efficiently share content on Wi-Fi Direct networks. As applications that enable content-sharing via Wi-Fi Direct are the most common type of application that directly use Wi-Fi Direct on mobile application stores, these novel methods would greatly improve the performance of the vast majority of existing Wi-Fi Direct applications. The power saving features of Wi-Fi Direct are reportedly improved by between 50-90% for group owner devices with the use of the two algorithms presented in [6]. The use of these algorithms would make the already power efficient Wi-Fi Direct standard even more appealing as a communication technology for enabling communication between devices over long stretches of time. Similarly, the scheme presented in [11] provides a way to manage Wi-Fi Direct's power consumption more efficiently. In a more recent study [5], the overhead of peer discovery in Wi-Fi Direct and the performance of its power saving features in practice was evaluated independently of any particular application. The security vulnerabilities of Wi-Fi Direct are exposed and presented in [17]. Besides research directly involving Wi-Fi Direct, many more studies exist

that delve into topics parallel to or related to this work.

This work seeks to explore an alternative method by which the less desirable effects of cellular connections can be mitigated. In contrast, many other works provide methods for directly improving cellular network performance. Descriptions of some of these endeavors are briefly presented here. The power inefficiency of cellular connections is of great concern to the research community. In [7], an algorithm was presented that significantly reduces the power consumption of LTE radios in mobile devices by intelligently switching between the Active and Idle radio states based on network traffic. For streaming traffic (video or otherwise), [14] similarly reduces the power consumption cellular radios with an algorithm that shapes streaming traffic into bursts. Work to improve the throughput of cellular networks has also been conducted. In [13], the benefits of multiple cores to manage LTE connections on a mobile device are explored. It was found that that using 2 cores instead of 1 and altering the memory management scheme could obtain significant improvements in throughput. The works previously described and others like them offer a wealth of improvements to communication over cellular networks. To the author's knowledge, even with these improvements, cellular communication would still suffer from fundamental latency and availability problems. It would also still be unable to match the power efficiency of Wi-Fi.

Chapter 4

Experimental Setup

The exact details of the conditions under which Super Tux Kart Direct was developed and tested are given below. Unless stated otherwise, it can be assumed that all experiments described below can be reproduced in roughly similar conditions.

4.1 Equipment

Super Tux Kart Direct was developed on two Samsung Galaxy Notes of the SGH-I717 variant[1]. These phablets[2] sport a dual-core 1.5 Ghz Scorpion CPU, an Adreno 220 GPU, 1 GB of RAM, LTE support, Wi-Fi Direct support, and more. Problems caused by the GPU lead to the development hardware being replaced by new hardware to run experiments. For some reason, the Adreno 220 GPU on the Samsung Galaxy Notes did not support hardware generation of mipmaps out of the box. As Super Tux Kart relies heavily on this functionality, the game would slow down to a crawl whenever the game was run on the Samsung Galaxy Notes as the Android system log would be continually flooded with error messages from the GPU at every iteration of Super Tux Kart. Furthermore, as a result of no mipmaps being generated, no textures could be loaded into the game world, and all the models would be some shade of gray during play. This state of affairs was tolerable during development, but could not be allowed to continue into experimentation. The slow down caused by the GPU was severe enough to obscure any performance differences that could have been observed when testing Super Tux Kart Direct with either Wi-Fi, Wi-Fi Direct,

[1]This version of the Samsung Galaxy Note is exclusive to the U.S. market.
[2]Phablets are smartphone/tablet hybrids

or LTE connections. Thus, the development equipment was replaced. Late in development, a fix was found that allowed the Samsung Galaxy Notes to run Super Tux Kart Direct correctly. An updated driver for the Adreno 220 GPU that add hardware generation of mipmaps functionality can be installed onto a Samsung Galaxy Note by loading version 10.1 of CyanogenMod, the open-source, custom Android firmware, onto a Samsung Galaxy Note. However, by the time this fix had been discovered, new equipment had already been obtained.

4.1.1 Experimental Equipment

All experiments performed during this work were done on Samsung Galaxy S4s of the SGH-I337 variant. These smartphones sport a quad-core 1.9 Ghz Snapdragon 600 CPU, an Adreno 320 GPU, 2 GB of RAM, LTE support, Wi-Fi Direct support, and more. The Samsung Galaxy S4 has none of the graphical problems that the Samsung Galaxy Note presented during development.

4.2 Experiments

A number of performance metrics were measured to properly test how well Super Tux Kart Direct performs when connections between players are enabled by either Wi-Fi, Wi-Fi Direct, or LTE. Each one of the experiments detailed below sought to find the value of either the overall Application Runtime, the response time to player inputs, the Player Input Throughput, or the Round Trip Network Latency under different conditions. Specifically, the number of players and the communication technology used varied according to which trial was being run for a particular experiment. Furthermore, each experiment was, for the most part, run independently of the others (i.e. only one value was ever being measured at one time). What follows is a detailed explanation of how each experiment was conducted.

All of the experiments described below were conducted under the following conditions. The experiments were conducted in a Boston residence with a single Wi-Fi access point shared by multiple people with varying numbers of Wi-Fi enabled devices connected to the access point at any given time. Thus, interference from other devices, doors, walls, ceilings, etc. for Wi-Fi and LTE should be assumed to be roughly similar to that of an average home full of tech savvy people. The Samsung Galaxy S4s used in the experiments were in

roughly the same place during each experiment and did not move. The phones were always within 1 meter of each other and within a fixed distance from the wireless access point. To the author, the use of an application enabled by either Wi-Fi, LTE, or Wi-Fi Direct in an environment similar to the one used in the experiments seems to be a common use case for each communication technology. If the effect of the experimental setup had a significant impact on the observed results of an experiment, it will be noted in the next chapter.

4.2.1 Application Runtime

Super Tux Kart Direct's runtime during a multiplayer session is a good indicator of how the player experience will vary when different technologies enable communication between devices. In this work, the runtimes of complete multiplayer races involving human players were not measured. Any multiplayer Super Tux Kart Direct race in which humans controlled the karts would have to factor in human reaction times and would severely limit repeatability. As accurate, meaningful measurements were desired, this method of gathering data was not used. Instead, Application Runtime was measured in this work as follows.

A special option called Auto Drive was added to Super Tux Kart Direct that alters the behavior of player karts for a brief period of time for the express purpose of taking measurements during multiplayer sessions. When this option is enabled, a player kart performs a simple, specific routine during a multiplayer session. Any player kart following this routine will wait at the starting line of a race for 60 iterations of the game loop, drive forward for 190 iterations of the game loop, and drive in reverse for 100 iterations of the game loop. In terms of what the player would see, this routine makes the player kart do the following. In the race track Amazonian Jungle, there is a river directly in front of the player karts some distance away at the beginning of the race. The Auto Drive routine makes the player kart move to the edge of this river, then back away slowly. The purpose of waiting 60 iterations to move at the start of the routine is to avoid the in-game penalty for trying to accelerate before a race has officially started. This penalty disables a kart's ability to move for a short period of time. The exact choice for what a player kart should do during the Auto Drive routine was completely arbitrary. All that was required was a short, simple, repeatable routine for evaluating the Application Runtime performance of Super Tux Kart Direct, and the routine given above satisfied that condition. The amount of time it takes for Super Tux Kart Direct to execute this simple routine is defined as the Application Runtime

in this work.

When both Auto Drive and the Application Runtime experiment are enabled, measurement of the Application Runtime begins after Super Tux Kart Direct has completed the initialization process for a multiplayer session and right before the first iteration of the game loop. Measurement of the Application Runtime ends when the Auto Drive routine ends. To ensure that these measurements are as accurate as possible, each one is taken at the microsecond scale. The Application Runtime experiment was conducted using either 2 phones or 4 phones depending on the trial. For each (number of phones, communication technology) pair, fifteen trials of the Application Runtime experiment were performed as described above. The median value of the results of all fifteen trials was presented as the observed Application Runtime for the (number of phones, communication technology) pair corresponding to that set of trials. This was done to remove the effects that a few outlying result values might have had on an average of the results.

4.2.2 Player Input Throughput

In a similar and related way to application runtime, the throughput of player inputs in a multiplayer session of Super Tux Kart Direct can be measured to determine the effect of different communication technologies on the player experience. In this work, Player Input Throughput was the only metric that was not measured as part of its own experiment. Instead, it is estimated from the results of the Application Runtime experiment.

As previously explained, the Auto Drive option controls a player kart in a multiplayer session of Super Tux Kart for 350 iterations of the game loop. During each iteration of the game loop, a player's input to Super Tux Kart Direct is sent to the game server and a response is received. A number of user inputs is processed at the game server and executed during each iteration of Super Tux Kart Direct equal to the number of players in the multiplayer session. Suppose n is the number of player karts in a multiplayer session of Super Tux Kart Direct. Then, the number of player inputs processed during the Auto Drive routine is given by $350n$. The throughput of player inputs for a particular trial of the application runtime experiment can be estimated by dividing $350n$ by the result of the trial. The value presented as the observed Player Input Throughput for a particular communication technology is defined as the median of all derived Player Input Throughputs for a particular set of trials. This was done to remove the effects that a few outlying result

values might have had on an average of the results.

4.2.3 Player Input Response Time

The Player Input Response Time is a measure of how quickly the Super Tux Kart Direct server responds to player inputs. More specifically, it is a measure of how long it takes, on average, for the server to send a response to a particular player's device after it has received that device's input for that iteration. It also illustrates the contribution of the Super Tux Kart game server to an individual device's application runtime.

In this work, Player Input Response Time was measured as follows. As previously described, both versions of the Super Tux Kart Direct server force the game to progress in lock-step. When the server obtains the input from a player's device during an iteration of a multiplayer session of Super Tux Kart Direct, it postpones sending out a response to that device until it has received the inputs of all other participating devices in that iteration of the multiplayer session. Similarly, each device will wait for a response from the server after sending out its input for that iteration before executing the next iteration of the game loop. Given the way the server executes, the Player Input Response Time for a particular player is defined as the elapsed time between when the Super Tux Kart Direct server receives a particular player's input and when the server sends out a response. As in previous experiments, the Auto Drive option was used in the Player Input Response Time experiment to ensure its repeatability. When both the Auto Drive option and the Player Input Response Time experiment are enabled, the Super Tux Kart Direct server records the Player Input Response Time for all players for every iteration of the Auto Drive routine. The average of the Player Input Response Times over all iterations of the Auto Drive routine for a particular player is defined as the observed Player Input Response Time for that player. In this work, the average of the observed Player Input Response Times for all players is defined as the observed Player Input Response Time for and is the proper result of a particular trial of the Player Input Response Time experiment. To ensure that this value was as accurate as possible, measurements were taken at the nanosecond scale.

The Player Input Response Time experiment was conducted using either 2 phones or 4 phones depending on the trial. For each (number of phones, communication technology) pair considered, fifteen trials of the Player Input Response Time experiment were performed as described above. The median value of the results of all fifteen trials was presented

as the observed Player Input Response Time for the (number of phones, communication technology) pair corresponding to that set of trials. This was done to remove the effects that a few outlying result values might have had on an average of the results.

4.2.4 Round Trip Network Latency

The Round Trip Network Latency is a good measure of how much overhead is incurred when communicating between devices. In this work, Round Trip Network Latency was measured as follows. Before a multiplayer session can begin, Super Tux Kart Direct must go through an initialization process where, among other things, a TCP connection to the game server is established. When the Round Trip Network Latency test is enabled, a small packet is sent to the game server immediately after a TCP connection to the game server is established. Super Tux Kart Direct will then wait to receive an acknowledgement packet from the server. The amount of time it takes for an acknowledgement packet to arrive after the initial packet is sent to the game server is used as an estimate for Round Trip Network Latency in this work. To ensure that this value was as accurate as possible, measurements were taken at the nanosecond scale. For each communication technology, fifteen trials of the Round Trip Network Latency test were performed as described above. The median value of the results of all fifteen trials was presented as the observed Round Trip Network Latency for the communication technology corresponding to that set of trials. This was done to remove the effects that a few outlying result values might have had on an average of the results.

Chapter 5

Results

In this chapter, the results of the experiments detailed in the previous chapter are presented. Possible explanations for the observed phenomenon will also be given after the results of each experiment.

5.1 Application Runtime

# of Phones	Wi-Fi	LTE	Wi-Fi Direct
2	49.2	62.4	33.0
4	52.3	54.9	36.6

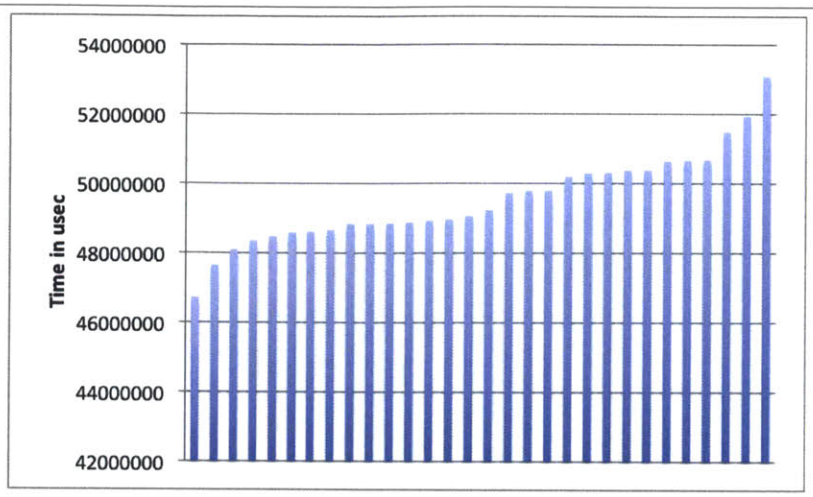
Table 5.1: Observed Application Runtime in seconds

As can be seen from Table 5.1, it would appear as though Wi-Fi Direct connections produce the shortest observed Application Runtime in Super Tux Kart Direct out of all the communication technologies tested. That Wi-Fi Direct should outperform Wi-Fi is an oddity. The reason for such low Application Runtimes will become clear when the results of the Round Trip Network Latency experiment are presented. The individual observations for each trial are shown in Figures 5-1 and 5-2.

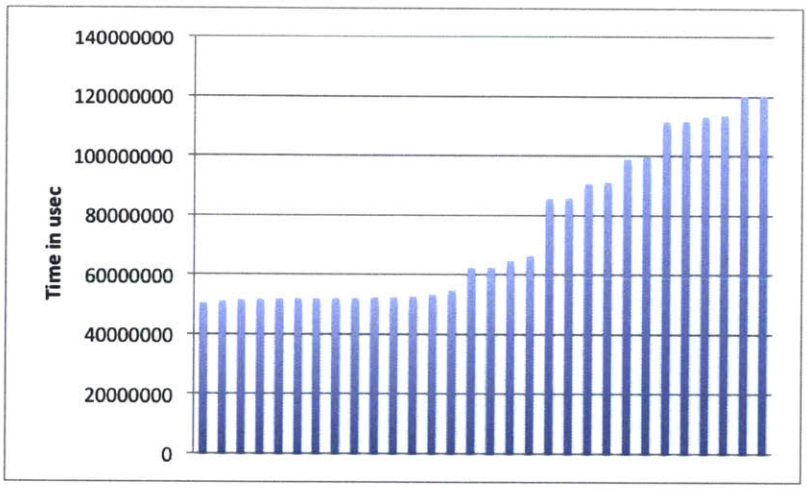
For the most part, the Application Runtime results for Wi-Fi and Wi-Fi Direct did not vary too widely during any particular set of trials. This is true with the notable exception of the results of one particular Wi-Fi Direct trial. During the set of trials in which Application

Runtime performance was being examined for multiplayer sessions of Super Tux Kart Direct involving 4 phones communicating over Wi-Fi Direct, there was a trial in which the Auto Drive routine took about twice as long to complete when compared to the other trials. During the trial, there were a few moments of extreme lag in the middle of the Auto Drive routine. This behavior was anomalous, and only observed during this particular trial in this work. Though Wi-Fi Direct performance is pretty consistent, it is important to note that momentary hiccups are still possible.

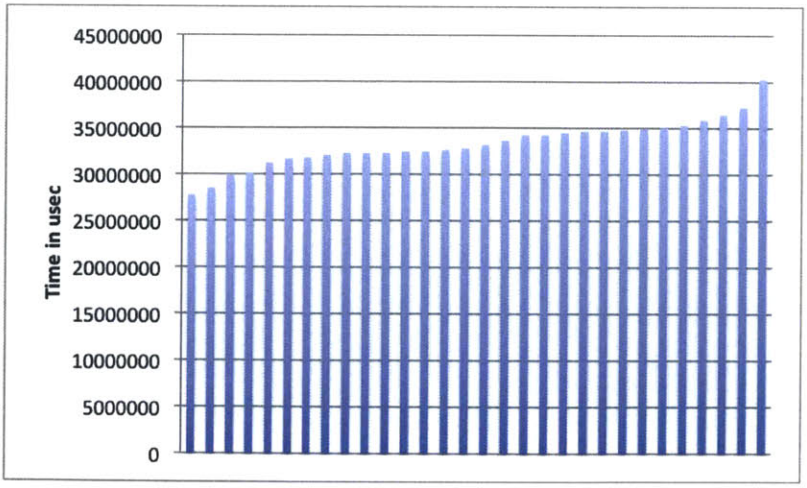
When LTE connections were used to enable multiplayer Super Tux Kart Direct sessions performance varied significantly depending on how many phones were being used during a set of trials. During the set of LTE trials in which only 2 phones were part of a multiplayer session, Application Runtime was, at times, almost double that observed during Wi-Fi Direct-enabled multiplayer sessions with the same number of phones. At other times, Application Runtime performance was comparable to that observed in Wi-Fi enabled sessions. In contrast, during the set of LTE trials in which 4 phones were part of a multiplayer session, performance in terms of Application Runtime was roughly comparable to that observed during Wi-Fi-enabled multiplayer sessions with the same number of phones. This disparity in performance between the 2-phone and 4-phone sets of trials could have been due to the nearest cellular tower temporarily servicing heavy network traffic during the 2-phone set of trials. It could also be that the packets being sent between devices and the server were just too small for LTE connections to handle efficiently at all times during the 2-phone set of trials.



(a) Wi-Fi

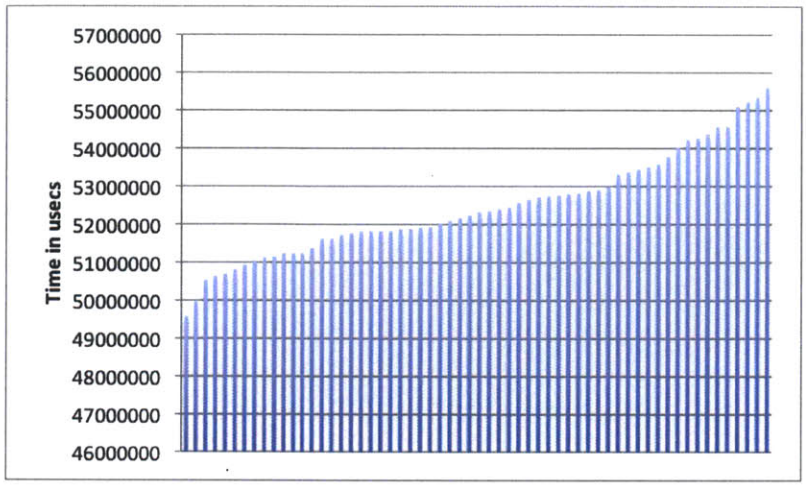


(b) LTE

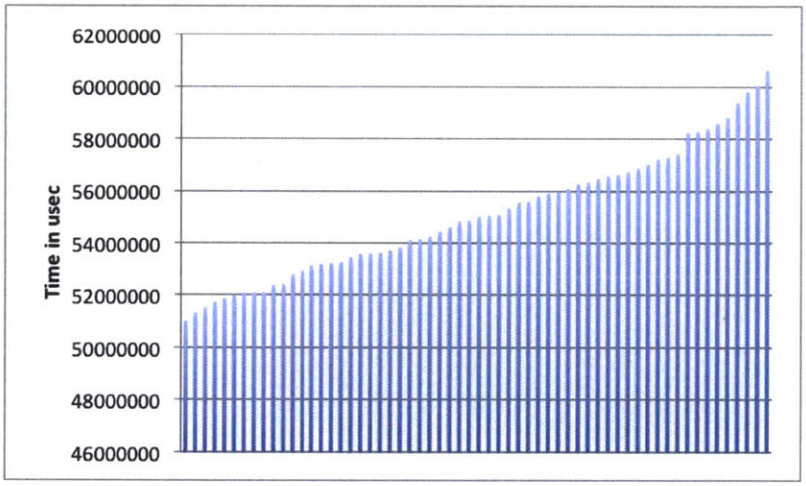


(c) Wi-Fi Direct

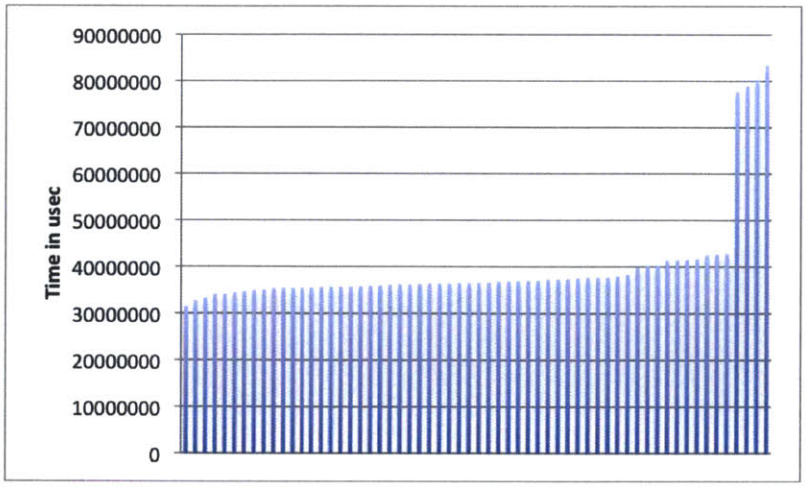
Figure 5-1: Application Runtime Results for 2 Phones



(a) Wi-Fi



(b) LTE



(c) Wi-Fi Direct

Figure 5-2: Application Runtime Results for 4 Phones

5.2 Player Input Throughput

# of Phones	Wi-Fi	LTE	Wi-Fi Direct
2	14.2	11.2	21.2
4	26.8	25.5	38.3

Table 5.2: Observed Player Input Throughput in inputs/second

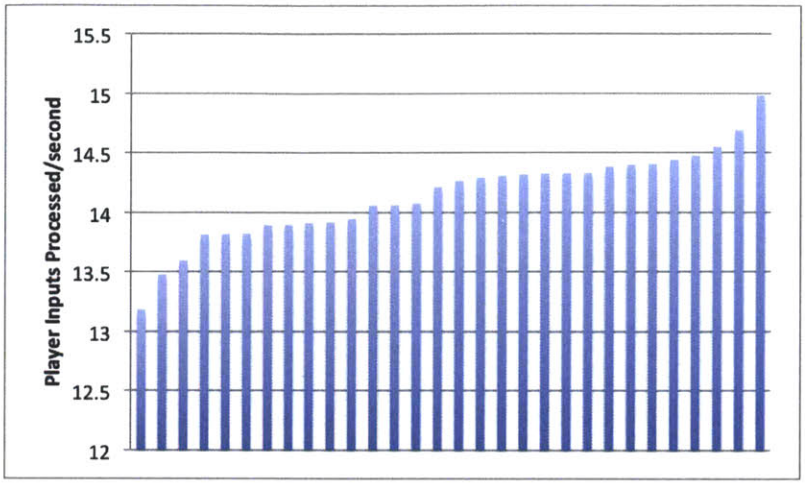
As can be seen from Table 5.2[1], it would appear as though the highest observed throughput in terms of player inputs is seen when communication in Super Tux Kart Direct is enabled by Wi-Fi Direct. The individual observations for each trial are shown in Figures 5-3 and 5-4.

Though Player Input Throughput is a simple transformation of Application Runtime, it offers a different perspective on how Super Tux Kart Direct performance is affected by the use of different communication technologies during multiplayer sessions. Instead of showing how performance is affected from the perspective of the player, Player Input Throughput provides insight on how performance is affected from the point of view of the game server. For example, the above table of observed Player Input Throughputs shows that, when a 2 phone, multiplayer Super Tux Kart Direct session is enabled by LTE, the game server is only able to process about half as many player inputs as when a 2 phone, multiplayer Super Tux Kart Direct session is enabled by Wi-Fi Direct. Thus, it can be seen that the choice of communication technology limits the performance of Super Tux Kart Direct on both the game server and all participating devices.

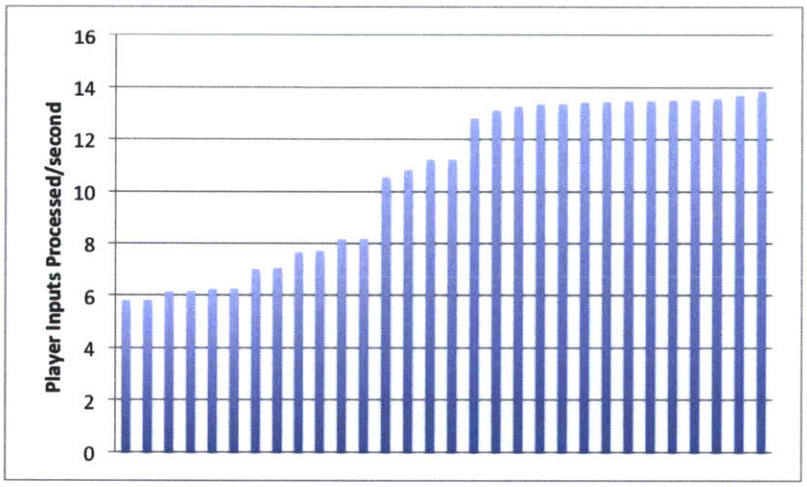
If not for the simple, lock-step design of the server, the differences between Wi-Fi, LTE, and Wi-Fi Direct would likely not be as stark. There are a number of well-known methods that have been developed over the years to mitigate the effects of slow connections in multiplayer games, and these methods could easily have been used to improve the performance of Super Tux Kart Direct on a player's device. Even when these methods are used, however, the difference between a slow connection and a fast connection in terms of performance and

[1]The number of iterations of Super Tux Kart Direct executed on average each second for a particular set of trials is equal to the observed Player Input Throughput divided by the number of phones used during that set of trials.

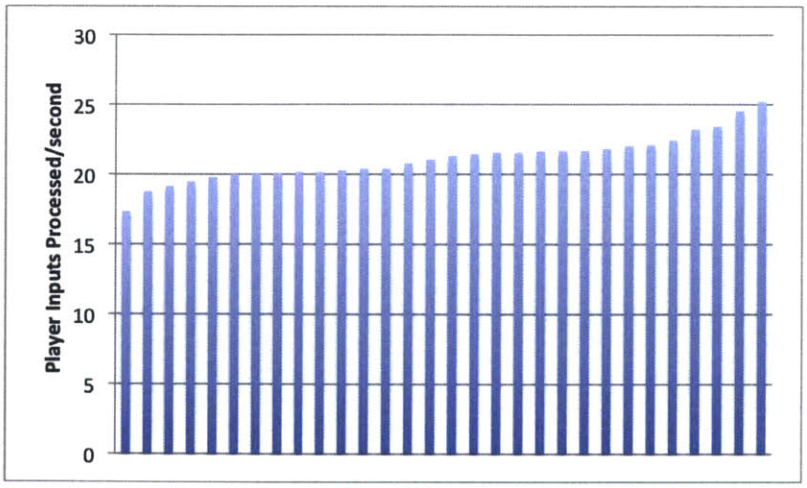
the perceived player experience is still significant. The simple design of the game server simply makes these differences more apparent.



(a) Wi-Fi

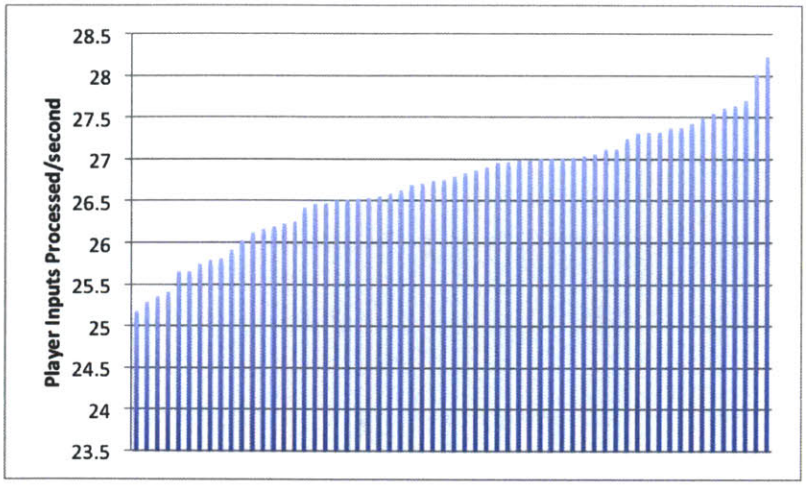


(b) LTE

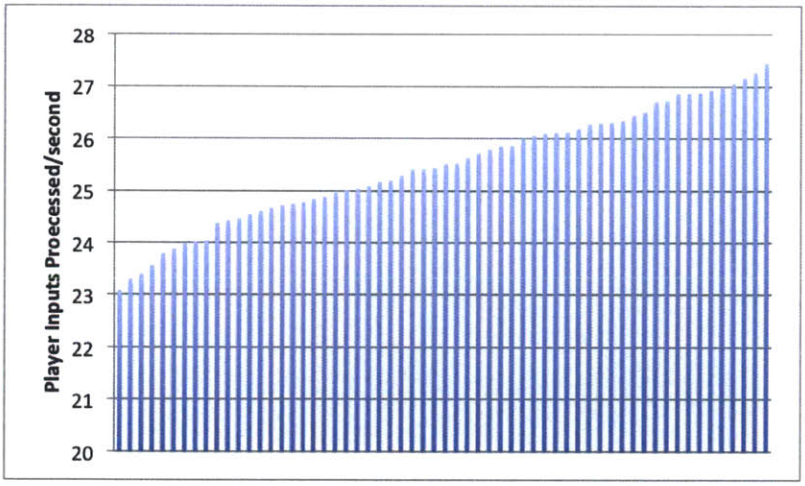


(c) Wi-Fi Direct

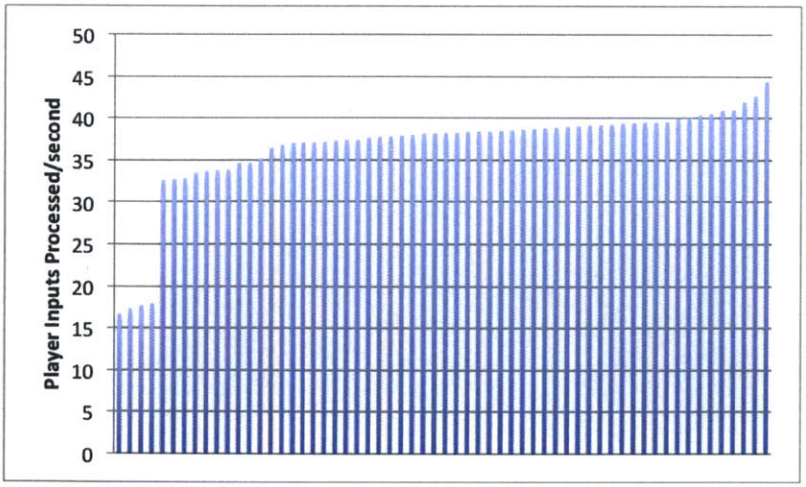
Figure 5-3: Player Input Throughput Estimates for 2 Phones



(a) Wi-Fi



(b) LTE



(c) Wi-Fi Direct

Figure 5-4: Player Input Throughput Estimates for 4 Phones

5.3 Player Input Response Time

# of Phones	Wi-Fi	LTE	Wi-Fi Direct
2	18.0	11.1	2.03
4	30.6	19.6	24.9

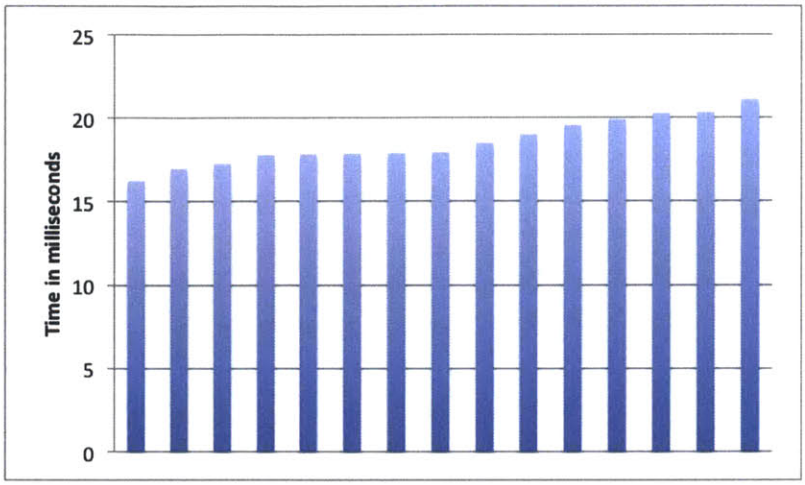
Table 5.3: Observed Player Input Response Time Results in milliseconds

As can be seen from Table 5.3, it would appear as though the lowest observed Player Input Response Times occur when communication in Super Tux Kart Direct is enabled by Wi-Fi Direct. The individual observations for each trial are shown in Figures 5-5 and 5-6.

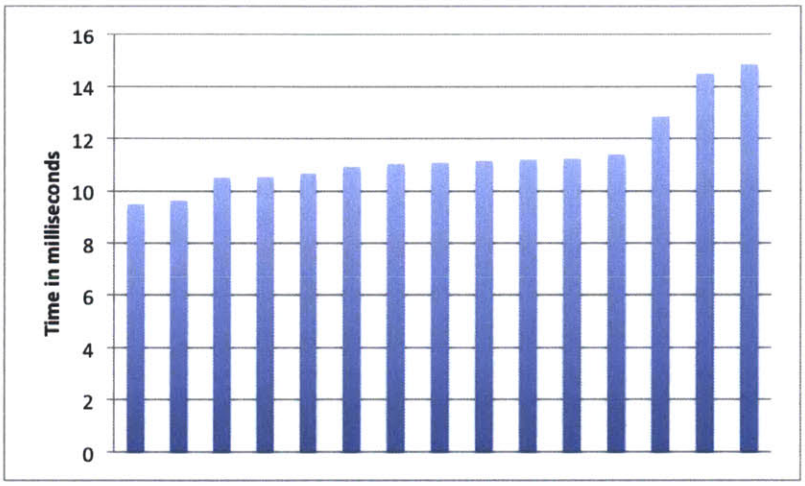
The low value observed during the set of trials in which 2 phones participated in a multiplayer Super Tux Kart Direct session enabled by Wi-Fi Direct was much lower than the other observed values for one simple reason. The Wi-Fi Direct version of the Super Tux Kart Direct game server will handle fewer connections than the Wi-Fi version of the game server for multiplayer sessions configured in exactly the same way. Due to the fact that the Wi-Fi Direct server runs on the same device as one of the client devices in a multiplayer session, one of the devices can communicate instantly with the server at all times in Wi-Fi Direct enabled multiplayer sessions. This effectively means that the Wi-Fi Direct server handles one connection fewer than the Wi-Fi server. In the 2-phone case, the server spends no time waiting to send back a response to all client devices because it is only waiting for the input of one device. This leads to a drastically low observed Player Input Response Time. In the 4-phone case, the overhead of having to wait for multiple player inputs becomes apparent in the observed Player Input Response Time for Wi-Fi Direct.

The low values observed during the LTE trials are not as easy to explain. It is possible that the fast response times of the Wi-Fi server were due to environmental conditions brought about by the time of day. Unlike the trials for Wi-Fi and Wi-Fi Direct, the trials for LTE in the 2 phone case were conducted very late at night (3:00-5:00). The cell tower for the residential area where the experiments were conducted may have had a much lighter load in terms of LTE connections that late at night when compared to its load during the day. This may have resulted in player inputs arriving at the Wi-Fi server consistently close together in terms of time. Such a thing would significantly reduce the response time of the

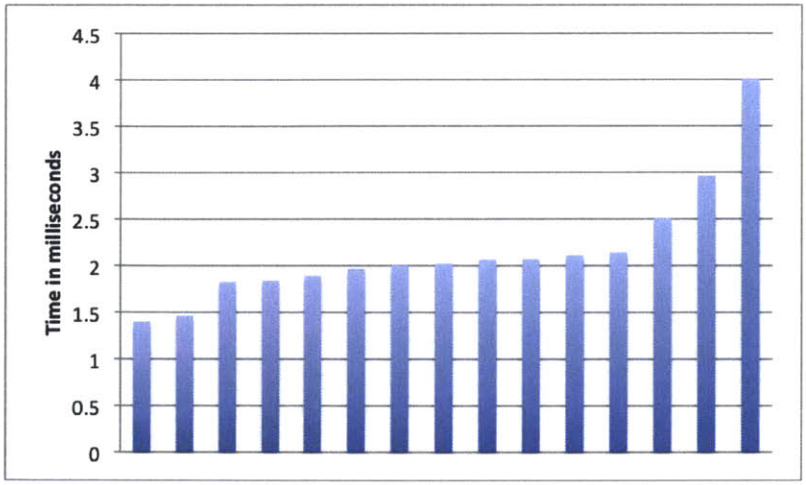
game server because the server would not have to wait as long on average to begin processing player inputs. Unfortunately, the explanation given above is simply conjecture based on an unexpected result. It could also simply be the case that LTE connections route data to a destination more consistently than Wi-Fi connections do. This seems likely, however.



(a) Wi-Fi

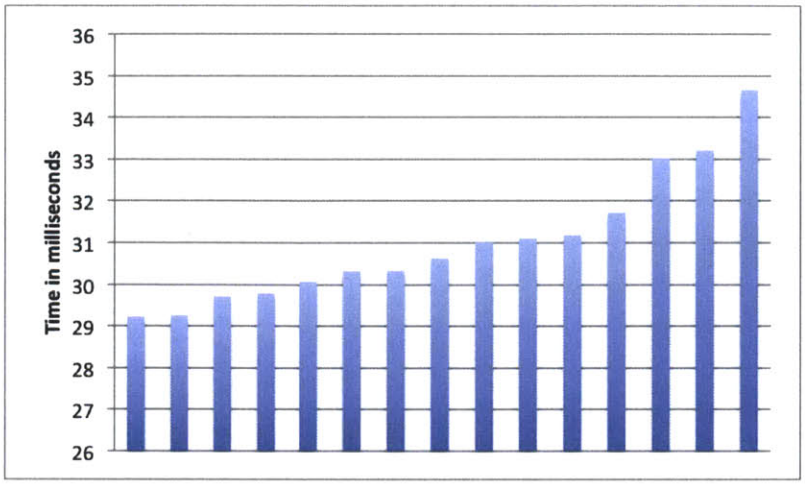


(b) LTE

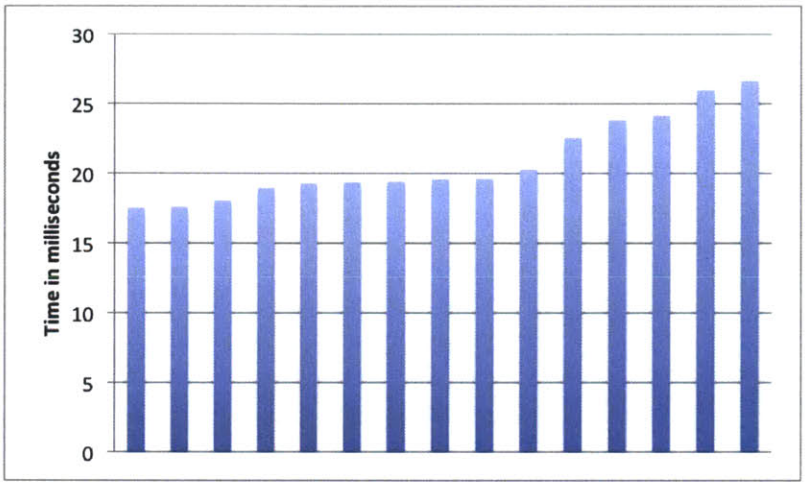


(c) Wi-Fi Direct

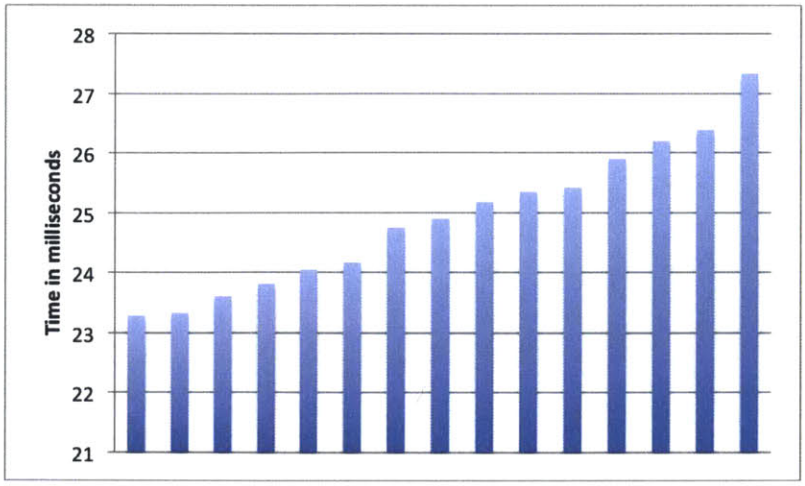
Figure 5-5: Player Input Response Time Results for 2 Phones



(a) Wi-Fi



(b) LTE



(c) Wi-Fi Direct

Figure 5-6: Player Input Response Time Results for 4 Phones

5.4 Round Trip Network Latency

# of Phones	Wi-Fi	LTE	Wi-Fi Direct
2	14.5	43.9	2.62
4	15.3	43.9	2.66

Table 5.4: Observed Round Trip Network Latency in milliseconds

As can be seen from Table 5.4, it would appear as though Wi-Fi Direct connections, on average, have the lowest observed Round Trip Network Latencies in Super Tux Kart Direct out of all the communication technologies tested. The individual observations for each trial are shown in Figures 5-7[2] and 5-8[3].

The low observed Round Trip Network Latency for Wi-Fi Direct was likely due to the close proximity of the phones during the experiment. The Wi-Fi access point in the residence where the experiment was conducted was roughly 5 meters below the phones on a lower floor during the experiment. In contrast, the phones were only physically 1 meter apart. Thus, during the experiment, Wi-Fi Direct communication physically travelled shorter distances through the air and travelled through fewer obstacles than Wi-Fi communication.

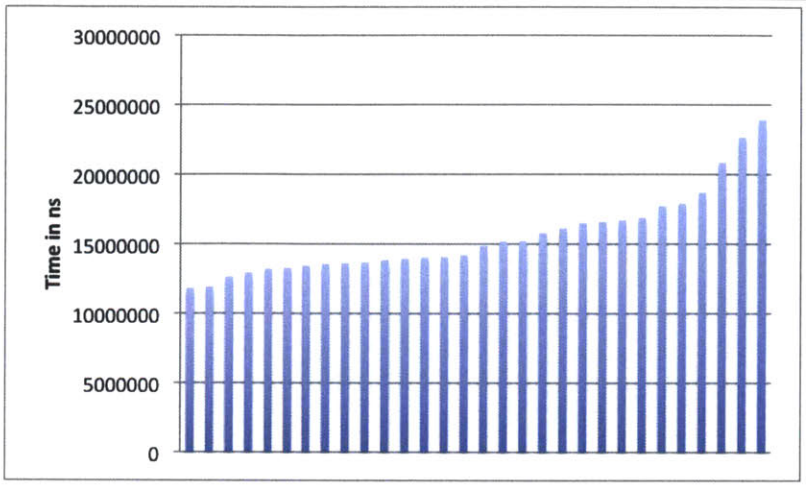
In a separate, mini-experiment, it was observed that the Round Trip Network Latency of Wi-Fi Direct increased by roughly a factor of 3 when the distance that the phones were kept apart was increased to 9 meters. This increased distance between phones was made in an open space without obstructions. It is likely that, given the results of the Round Trip Network Latency experiment and the previously mentioned mini-experiment, the Round Trip Network Latency observed during a multiplayer session of Super Tux Kart is roughly the same when either Wi-Fi or Wi-Fi Direct enable communication given one condition. If devices in the network are the same distance away from the device acting as the access point in both cases, the results of the experiments suggest that roughly the same performance in terms of Round Trip Network Latency will be observed. This behavior would explain why Wi-Fi Direct performed so much better than the other communication technologies in

[2]Note that there are fewer Wi-Fi Direct observations. This is due to the fact that one of the phones acts as both access point and client. Thus, it does not make any server connections. It simply interacts directly with the server when acting as a client.

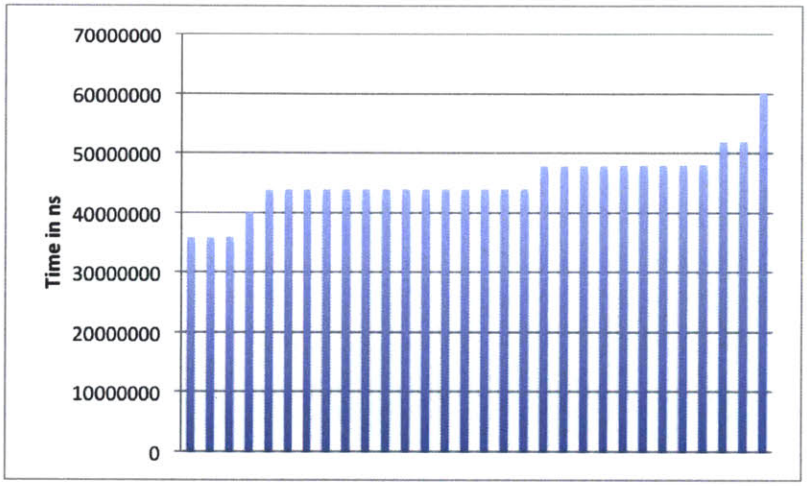
[3]Ibid

previous experiments.

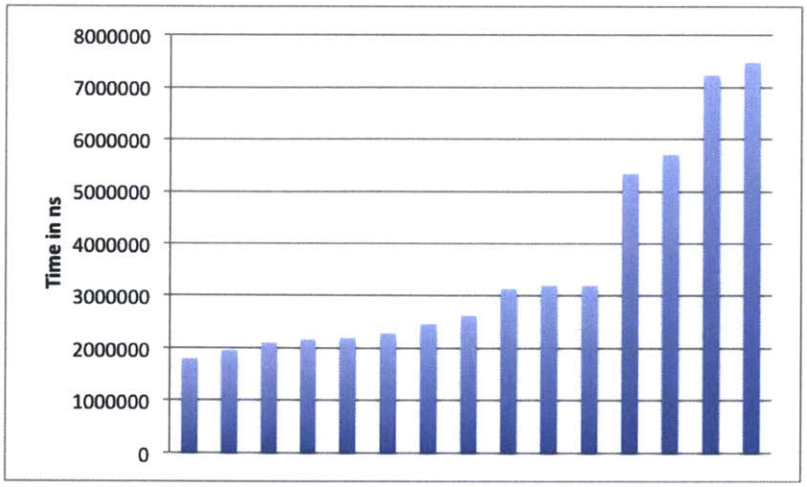
The observed Round Trip Network Latencies for LTE connections in Super Tux Kart Direct were significantly larger than those observed for both Wi-Fi and Wi-Fi Direct. The results of the experiment show that, though cellular network performance has greatly improved over its predecessors, LTE connections are not yet able to provide service with latencies or power efficiency as low as that of Wi-Fi. Wi-Fi Direct, however, seems fully capable of doing this in situations where locality can be leveraged.



(a) Wi-Fi

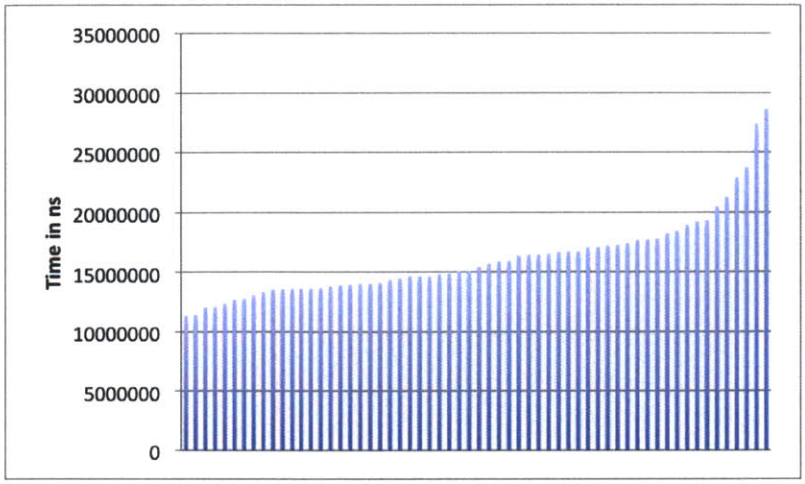


(b) LTE

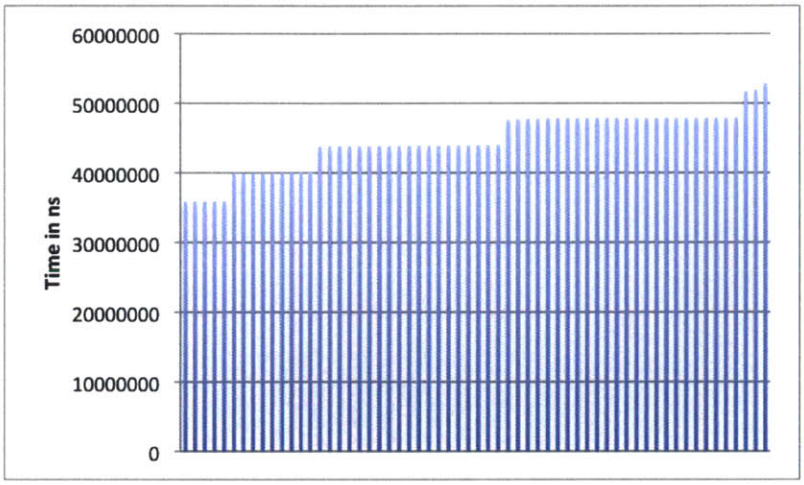


(c) Wi-Fi Direct

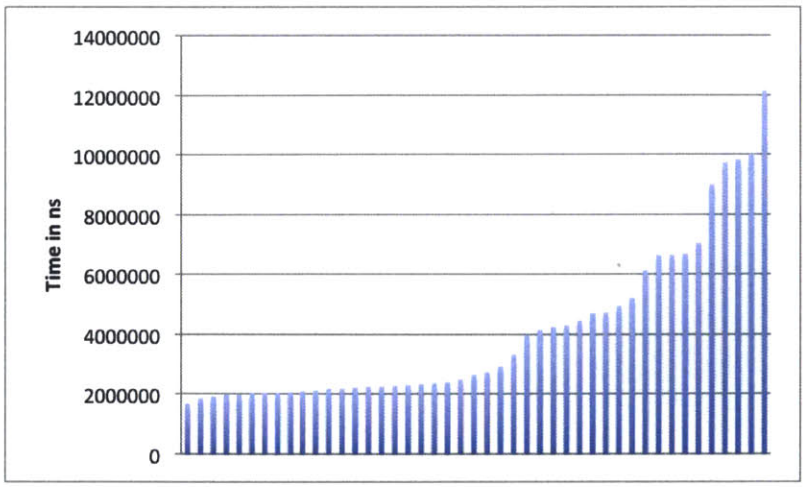
Figure 5-7: Round Trip Network Latency Results for 2 Phones



(a) Wi-Fi



(b) LTE



(c) Wi-Fi Direct

Figure 5-8: Round Trip Network Latency Results for 4 Phones

5.5 Performance Breakdown

Figures 5-9 and 5-10 show the previously presented Application Runtimes divided into 3 distinct runtimes: Server Runtime, Networking Time, and Client Runtime. With respect to the figures, Server Runtime is defined as the percentage of the Application Runtime that was spent processing player inputs at the game server. This percentage was calculated from the observed Player Input Response Times. By converting the observed Player Input Response Times into seconds and multiplying them by the number of iterations in the Auto Drive routine, estimates for the number of seconds of the Application Runtime that is spent executing code on the game server can be obtained. These estimates divided by the Application Runtime yielded the Server Runtime. Similarly, the Networking Time is defined as the percentage of the Application Runtime that was spent transmitting data via Wi-Fi, Wi-Fi Direct, or LTE and is calculated from the observed Round Trip Network Latencies. Lastly, the Client Runtime is defined as the percentage of the Application Runtime that was spent executing the Super Tux Kart Direct game code on player devices. This value was calculated by subtracting the Server Runtime and Networking Time from 100 percent. In short, the figures below show how time was spent when executing Super Tux Kart in different configurations.

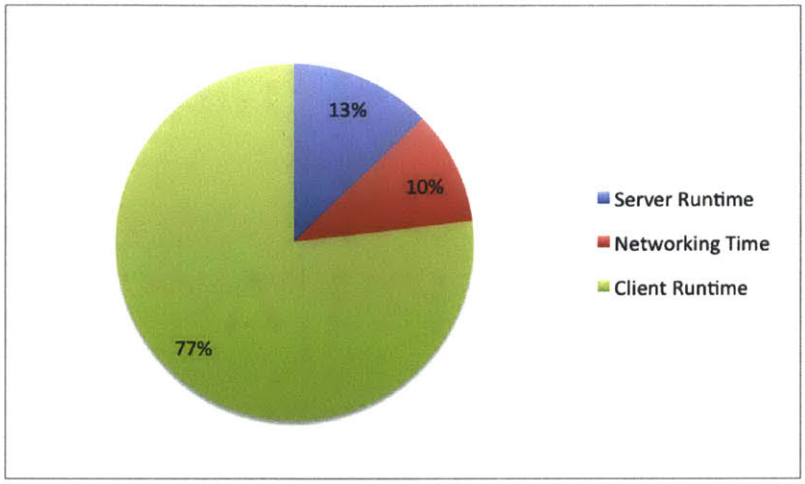
As discussed in the Design chapter, the Super Tux Kart Direct code executing on a player device will block the execution of the next iteration of the game until it has received a response from the server. The percentage of time that Super Tux Kart Direct spends blocking in this fashion can be calculated by adding the Server Runtime percentage and the Networking Time percentage together.

During the sets of trials for the Application Runtime experiment in which LTE connections were used to enable multiplayer sessions of Super Tux Kart, the percentage of time spent blocking game progress was much higher compared to other configurations. In the set of trials for the Application Runtime experiment in which 4 phones communicated over an LTE network, 41% of the Application Runtime on average was spent blocking game progress. This was mostly due to the high observed Round Trip Network Latencies for LTE enabled Super Tux Kart Direct multiplayer sessions. In other words, between Server Time and Networking Time, Networking Time contributed the most to the large amount of time spent blocking game progress during the LTE enabled sets of trials. Roughly a quarter of

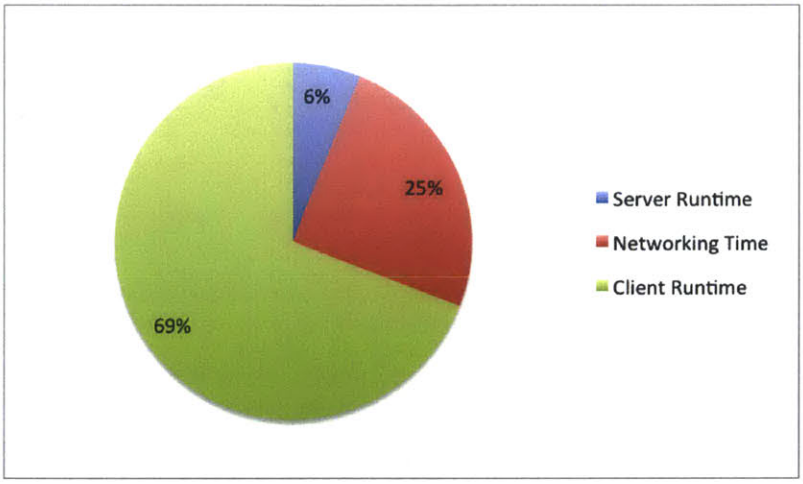
the Application Runtime was spent sending data over the LTE network during LTE enabled sets of trials. When the number of participating devices in a set of trials for the Application Runtime experiment increased, the Networking Time for the LTE configuration would increase even more.

Interestingly, the greatest contributing factor to the percentage of time spent blocking game progress in configurations in which either Wi-Fi or Wi-Fi Direct enabled multiplayer sessions was the high observed Player Input Response Time[4]. In other words, between Server Time and Networking Time, Server Time contributed the most to the amount of time spent blocking game progress. While the percentage of Application Runtime composed of Networking Time remained the same for Wi-Fi and Wi-Fi Direct enabled sets of trials when the number of phones was increased, the Server Time increased significantly with an increase in the number of phones in the Wi-Fi and Wi-Fi Direct enabled sets of trials. Initial findings seem to point to a need for reduced network latencies in LTE networks for better performance in Super Tux Kart Direct. Additonally, these findings seem to indicate a need for the Super Tux Kart Direct server code to be optimized to better handle Wi-Fi and Wi-Fi Direct configurations.

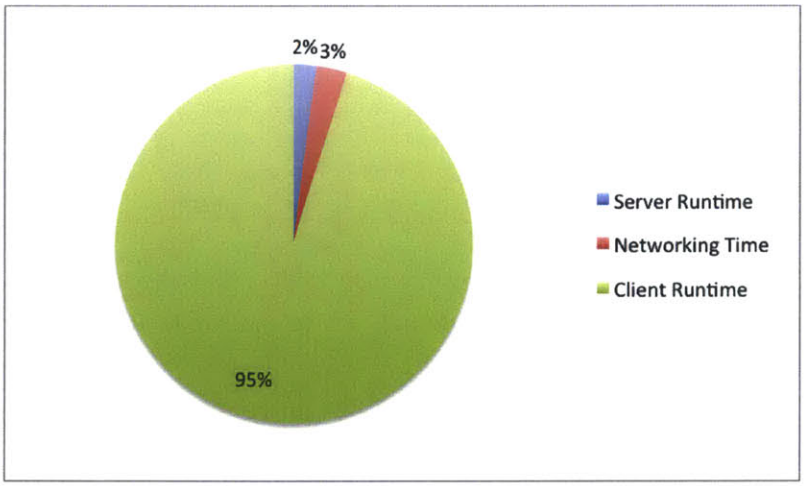
[4]This is true with the exception of the Wi-Fi Direct set of trials in which only 2 phones participated. The optimization of having the Super Tux Kart Direct game code communicate directly with the game server leads to very low Player Input Response Times in Wi-Fi Direct configurations.



(a) Wi-Fi

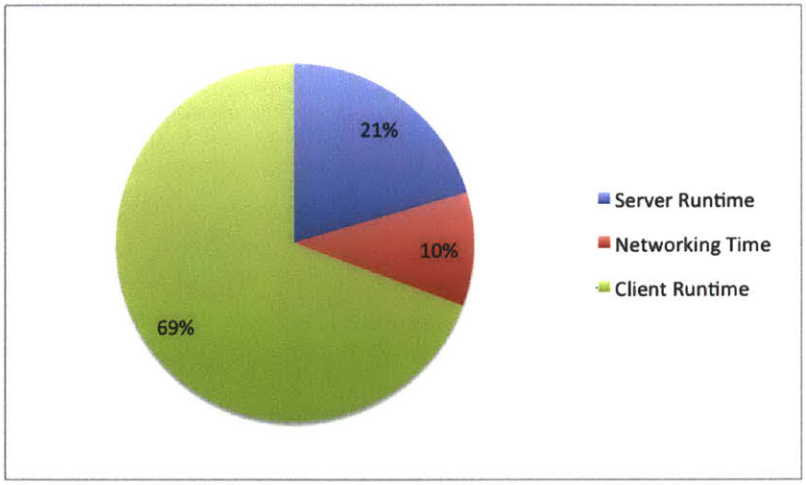


(b) LTE

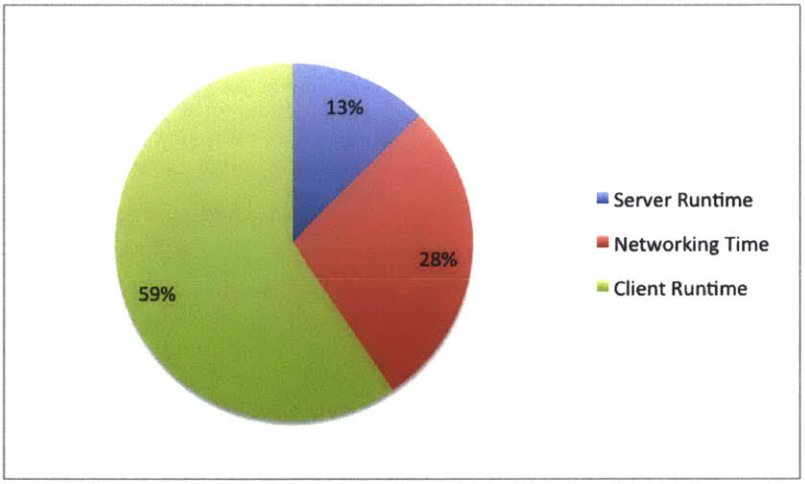


(c) Wi-Fi Direct

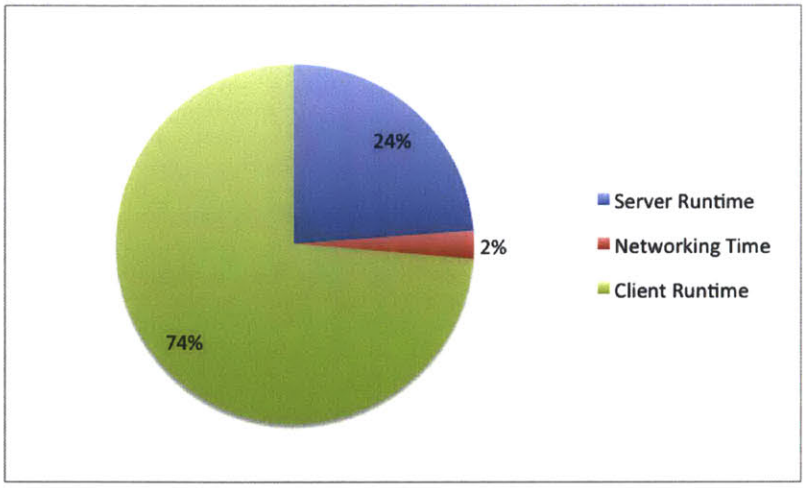
Figure 5-9: Performance Breakdown for 2 Phones



(a) Wi-Fi



(b) LTE



(c) Wi-Fi Direct

Figure 5-10: Performance Breakdown for 4 Phones

Chapter 6

Conclusions

In this work, the benefits of using Wi-Fi Direct to enable communication between devices instead of cellular connections were explored. With initial experiments comparing the performance of multiplayer sessions of Super Tux Kart Direct enabled by Wi-Fi, LTE, and Wi-Fi Direct with varying numbers of phones, it seems that Wi-Fi Direct was, in many respects, the best technology for enabling multiplayer play in the proof of concept system. It proved to be the technology that allowed Super Tux Kart Direct to deliver the fastest response time during multiplayer sessions. It also happens to be just as power efficient as Wi-Fi. Though the author envisions many uses for Wi-Fi Direct in future applications, Wi-Fi Direct is not suited for use in all applications.

Super Tux Kart Direct has certain properties that make Wi-Fi Direct a viable choice as a communication technology. First, the system did not heavily rely on any external resources to advance execution. In fact, all resources needed to run the game were on each device. If Super Tux Kart Direct had needed to use some distant, external resource (e.g. web service, remote server for performing big calculations), it would have had to rely on Wi-Fi or LTE to communicate with such resources. For applications with such dependencies, using Wi-Fi or LTE to enable communication is likely the best solution. Second, the amount of extra work performed by the group owner when Wi-Fi Direct enabled multiplayer sessions of Super Tux Kart Direct was kept to a minimum. One of the benefits of having designed the game server so simply is that it had a very lightweight implementation that did not overly affect the execution of the Super Tux Kart game code on the group owner device. For applications in which the group owner must do a lot extra work on top of an already

intensive application, Wi-Fi Direct might not be a good option. The extra work done by the group owner might slow execution down to a crawl in the aforementioned case. That said, there are many types of mobile applications that would either benefit from or could be enabled by the use of Wi-Fi Direct.

As previously stated games are one of the most common and popular types of applications in mobile app stores. Many mobile games of certain genres lack the online multiplayer functionality that games of the same genre would have on a console or personal computer. These games tend to belong to genres where fast, reliable communication technology is required to enable multiplayer play (e.g. first person shooters, racing games, etc.). Of those in app stores that offer online multiplayer functionality, many of them require players to remain in a location with a Wi-Fi connection. Cellular connections often do not offer good enough network performance to enable multiplayer play for these fast-paced, multiplayer games, and quickly drain the battery, thus, severely limiting the length of play. Wi-Fi Direct could enable these games to be played by groups of people on the go with its high speed, low latency connections, however. It could also enable developers to implement modes of play previously unseen in mobile games. For example, games that offer co-op modes[1] of play, an almost unheard of mode of play for a mobile game, would be quite viable if Wi-Fi Direct were used to enable communication. The low-latency, power efficient Wi-Fi connections between a small number of players would ensure that players experience the same, changing game world together for extended periods of time with no lag. Cooperative experiences offered by games like Portal 2 or Left 4 Dead could be possible. Other applications could benefit from the use of Wi-Fi Direct, as well.

Though perhaps less obvious than enabling fast-paced, multiplayer games for mobile users, Wi-Fi Direct could also be used for such applications as video streaming or secure communication. In the case of video streaming, Wi-Fi Direct could enable a family on a road trip to more efficiently stream a movie/video to all Wi-Fi enabled devices in a car. Conceivably, a Wi-Fi Direct group owner could be downloading and displaying chunks of a video stream from a service like YouTube or Netflix via a cellular connection and

[1]A true co-op mode of play is not to be confused with the minor interactions offered to players in mobile games who register their game on social media. In those kinds of interactions, a player's friend can periodically send in-game currency or unlock new experiences by their actions on social media. In a true co-op mode of play, a small number of players team up to directly experience a game's single player scenarios together. This usually takes the form of real-time cooperation in which players control separate avatars to advance through what would otherwise be the single player experience of a game.

simultaneously forwarding those same video chunks to all devices connected it via Wi-Fi Direct. In this way, the non-group owner devices conserve battery charge and all users are able to enjoy the movie/video. In the case of secure communication, communication over Wi-Fi Direct has the potential to provide more privacy than communication over Wi-Fi or LTE. Unlike Wi-Fi or LTE networks, the number of machines that sensitive data must travel through to reach its destination is at most 1 in a Wi-Fi Direct network.[2] Thus, the likelihood of sensitive data being recorded by an intermediary machine taken over by a malicious agent is reduced significantly. Furthermore, all connections are trusted in a Wi-Fi Direct Network by virtue of the fact that the user of a Wi-Fi Direct enabled device must acknowledge and accept any attempts to form a connection to the user's device. If end-to-end encryption is also used, sensitive communications become even more difficult for an attacker to intercept via techniques like air-sniffing. It is the author's belief that many more applications for Wi-Fi Direct exist that were not mentioned above. Perhaps more will become apparent when future work for this project is conducted.

6.1 Future Work

Some work still remains to be done with respect to this thesis. Due to time constraints or the availability of new technology, some experiments were not conducted in this work. Furthermore, the design of Super Tux Kart Direct did not reflect current practices in multiplayer games for hiding lag from the players. Additional experiments and additional improvements to the proof of concept system would better show what applications would benefit from Wi-Fi Direct communication over cellular communication. What follows is a discussion of the design improvements and additional experiments the author will incorporate into future work.

In fast-paced multiplayer games, often some kind of prediction is used to hide the effects of a poor connection to players in current games. Whether it be prediction on the client side or the server side, the future state of the game is predicted according to some algorithm to allow each player's game state to progress without having to wait for data from the server at every iteration. When conflicts between the real game state and the predicted game state on a player device occur, some kind of correction is done. It is during these corrections that

[2]Communications between devices might have to use the group owner as an intermediary.

a player will perceive lag. The current design of Super Tux Kart Direct does not hide the effects of poor connections at all. The lock-step design ensures that the player with the slowest connection to the game server will determine the speed of a multiplayer session. To better show the effects of using Wi-Fi Direct instead of a cellular connection in a modern, fast-paced, multiplayer game, the design of Super Tux Kart Direct will be updated in future work to make use of kart movement prediction. In future work, the performance of the old design will be compared to the updated design by comparing the results of the Application Runtime, Player Input Throughput, Player Input Response Time, and Round Trip Network Latency experiments for both designs.

It is clear from this work and the work of others, that network communication overhead increases substantially when the number of devices connected over Wi-Fi Direct increases. Some studies have claimed that Wi-Fi Direct performance becomes intractable after 5 or 6 devices connect to each other. It would be interesting and useful to quantify the network overhead in Super Tux Kart Direct when more than 4 devices engage in a multiplayer session. In future work, the Application Runtime, Player Input Throughput, Player Input Response Time, and Round Trip Network Latency experiments involving 8 or more phones will be conducted.

The cellular standards presented in this work are actively being improved upon and will soon be updated. Instead of being pre-4G technologies, the updates to the Mobile WiMAX and LTE standards promise to bring true 4G standards to the global market. The Mobile WiMAX 2 standard has already been submitted to the ITU for IMT-Advanced standardization (i.e. true 4G standardization), and LTE-Advanced, the true 4G update to the pre-4G LTE standard, has already been implemented and deployed in Russia and Korea. It would be interesting and useful to quantify the performance of true 4G communication in Super Tux Kart Direct, and to compare it to Wi-Fi and Wi-Fi Direct performance. In future work, the Application Runtime, Player Input Throughput, Player Input Response Time, and Round Trip Network Latency experiments will be conducted using true 4G technologies.

In the near future, the current set of 802.11 standards will be enhanced by the addition of the 802.11ac and 802.11ad standards. These standards would enable gigabit bandwidth communication at current Wi-Fi power efficiency for any device that supported them. This technology would substantially boost the performance of Wi-Fi Direct as well. It would be

useful and interesting to compare the performance of the next iteration of cellular communication to the next iteration of Wi-Fi. Thus, in future work, the Application Runtime, Player Input Throughput, Player Input Response Time, and Round Trip Network Latency experiments for Wi-Fi and Wi-Fi Direct will be conducted using these new standards.

Some vital metrics were not measured in this work. Unlike with Wi-Fi, devices communicating over Wi-Fi Direct or LTE can move about much more freely. The effects of mobility on Super Tux Kart Direct performance for Wi-Fi Direct and LTE connections was not explored in this thesis. Thus, in future work, the Application Runtime experiment for 2, 4, and, possibly, 8 phones will be conducted on moving buses and subway trains to evaluate these effects. Though, a mini-experiment was performed as a part of the Round Trip Network Latency experiment, the effects of the distance between Wi-Fi Direct devices on Super Tux Kart Direct performance was not formally measured. In future work, the effects of distance on Super Tux Kart Direct performance will be measured by conducting the Application Runtime, Player Input Throughput, Player Input Response Time, and Round Trip Network Latency experiments for Wi-Fi Direct at distances of 1 meter, 3 meters, and 10 meters. The power consumed when running Super Tux Kart Direct under various configurations was also not measured. To provide further evidence of the power efficiency of Wi-Fi Direct, a new experiment for 2, 4, and 8 phones will be conducted. This experiment will likely involve enabling the Auto Drive option in Super Tux Kart Direct in a desired configuration and using power monitoring tools to measure a player device's power consumption.

To summarize, the following experiments will be conducted in future work. All experiments conducted in this work will be conducted again using a more modern Super Tux Kart Direct design. The Application Runtime, Player Input Throughput, Player Input Response Time, and Round Trip Network Latency experiments conducted in this work will be conducted again with 8 phones and, possibly, 16 phones. Also, all new experiments will be conducted with 8 phones, and, possibly, 16 phones. If the technologies are deployed swiftly enough, trials for the Application Runtime, Player Input Throughput, Player Input Response Time, and Round Trip Network Latency experiments will be conducted using either Mobile WiMAX 2, LTE-Advanced, 802.11ac or 802.11ad connections. Application Runtime experiments will be conducted on subway trains and buses to evaluate the effects of mobility on Super Tux Kart Direct games enabled by either Wi-Fi Direct or LTE. Ap-

plication Runtime experiments will also be conducted at varying distances to evaluate the effects of distance on the performance of Super Tux Kart Direct games enabled by Wi-Fi Direct. Finally, experiments will be conducted to measure the power consumption of player devices running Super Tux Kart Direct in various configurations.

Bibliography

[1] IEEE Standards Association. `http://grouper.ieee.org/groups/`.

[2] Supper Tux Kart. `http://supertuxkart.sourceforge.net/`.

[3] Supper Tux Kart - Xapantu Port. `http://supertuxkart.sourceforge.net/User:Xapantu/Android`.

[4] Wi-Fi Alliance. `http://www.wi-fi.org/`.

[5] Daniel Camps-Mur, Andres Garcia-Saavedra, and Pablo Serrano. Device-to-device communications with WiFi Direct: overview and experimentation. *Wireless Communications, IEEE (Volume:20, Issue:3)*, 2013.

[6] Daniel Camps-Mur, Xavier Perez-Costa, and Sebastia Sallent-Ribes. Designing Energy Efficient Access Points with Wi-Fi Direct. *Computer Networks (Volume:15, Issue:13)*, 2011.

[7] Shuo Deng and Hari Balakrishnan. Traffic-Aware Techniques to Reduce 3G/LTE Wireless Energy Consumption. In *Proceedings of 8th International Conference on Emerging Networking Experiments and Technologies*, 2012.

[8] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck. A Close Examination of Performance and Power Characteristics of 4G LTE Networks. In *Proceedings of MobiSys 2012*, 2012.

[9] International Telecommunication Union. The Wold in 2013: ICT Facts and Figures, 2013.

[10] Teemu Karkkainen, Mikko Pitkanen, and Jorg Ott. Enabling ad-hoc-style communication in public WLAN hot-spots. *ACM SIGMOBILE Mobile Computing and Communications Review (Volume:17, Issue:1)*, 2013.

[11] Keun-Woo Lim, Woo-Sung Jung, Hanna Kim, Jina Han, and Young-Bae Ko. Enuanced Powe Management for Wi-Fi Direct. In *Wireless Communications and Networking Conference (WCNC)*, 2013.

[12] Tuan Nguyen Duong, Ngoc-Thanh Dinh, and YoungHan Kim. Content Sharing using P2PSIP Protocol in Wi-Fi Direct Networks. In *Proceedings of 4th International Conference on Communications and Electronics (ICCE)*, 2012.

[13] Anas Showk, Felix Bruns, Sebastian Hessel, Attila Bilgic, and Irv Badr. Optimal Resource Management for a Model Driven LTE Protocol Stack on a Multicore Platform. In *Proceedings of the 8th ACM International Workshop on Mobility Management and Wireless Access*, 2010.

[14] Matti Siekkinen, Mohammed Ashraful, and Mika Aalto. Streaming over 3G and LTE - How to Save Smartphone Energy in Radio Access Network-Friendly Way. In *Proceedings of the 5th Workshop on Mobile Video*, 2013.

[15] Joshua Wright. Dispelling Common Bluetooth Misconceptions. `http://www.sans.edu/research/security-laboratory/article/bluetooth`, 2007.

[16] Hayoung Yoon and JongWon Kim. Ccollaborative Streaming-based Media Content Sharing in WiFi-enabled Home Networks. *IEEE Transactions on Consumer Electronics (Volume:56, Issue:4)*, 2010.

[17] Seokung Yoon, SoonTai Park, Haeryoung Park, and Hyeong Seon Yoo. Security Analysis of Vulnerable Wi-Fi Direct. In *Proceedings of 8th International Conference on Computing and Networking Technology (ICCNT)*, 2012.