Final thesis

A study on Android games: 3G energy consumption, CPU-utilization and system calls

by

Mathias Almquist & Viktor Almquist

LIU-IDA/LITH-EX-G--15/004--SE
2015-03-09
Upphovsrätt

Detta dokument hålls tillgängligt på Internet – eller dess framtida ersättare – från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.

 Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida http://www.ep.liu.se/

Copyright

The publishers will keep this document online on the Internet – or its possible replacement – from the date of publication barring exceptional circumstances.

The online availability of the document implies permanent permission for anyone to read, to download, or to print out single copies for his/hers own use and to use it unchanged for non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional upon the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its www home page: http://www.ep.liu.se/.

© Mathias Almquist & Viktor Almquist
Examensarbete

A study on Android games: 3G energy consumption, CPU-utilization and system calls

av

Mathias Almquist & Viktor Almquist

LIU-IDA/LITH-EX-G--15/004--SE

2015-03-09

Handledare: Ekhiotz Jon Vergara
Examinator: Simin Nadjm-Tehrani
Abstract

The popularity of mobile games has increased drastically during the recent years and many people use them as their main source of entertainment. Mobile games communicate with other devices over the network which consumes a lot of energy, especially when connected to cellular networks (e.g., 3G). This high energy expense can feel unjustified to the player since always-on network connectivity is not required in order to play most games.

Furthermore, the number of malware-infected applications in official application stores has increased significantly in the recent years. These malware-infected applications can gain unrestricted access and control of users phones which can be a threat to security. Information about the behaviour characteristics of games can be used to develop or improve systems for detecting malware applications.

In this thesis, 20 popular Android games are analysed with a focus on the data communication, CPU utilization and system call behaviour. The main subject of the data communication study is the 3G communication energy consumed by games. The system call study aims at quantifying the number and type of calls used by games. This may be useful in a further study of harmful behaviour by apps.

The profiling results presented in this report show that the communication energy varies drastically among games. Games with a very similar gameplay can consume very different amounts of energy which indicates that there is room for improvements in many of the games. Ad-free games consume significantly less energy than games that use in-app advertisements. The results show that improving the advertisement fetching policy could reduce the energy consumption of these games. The majority of the games can be played without network connectivity and therefore the communication energy consumed could be completely avoided. The thesis also shows that games use a wide variety of system calls and that many of the system calls are common among the games.
## Contents

1 Introduction ................................................................. 1
   1.1 Context and Motivation ............................................. 1
   1.2 Goals ........................................................................ 2
   1.3 Approach ................................................................. 2
   1.4 Limitations ............................................................... 4
   1.5 Indended Audience .................................................... 4
   1.6 Related Scientific Work ............................................. 4
      1.6.1 Energy-Related Studies ..................................... 4
      1.6.2 Profiling-Related Studies .................................. 5
   1.7 Report Outline ........................................................ 6

2 Background ................................................................. 7
   2.1 3G Energy Consumption ........................................... 7
   2.2 Description of Used Tools ......................................... 9
      2.2.1 Monkey Gamer .................................................. 9
      2.2.2 Energybox .......................................................... 9
      2.2.3 Tcpdump ............................................................ 10
      2.2.4 Strace ................................................................. 10
      2.2.5 Top ..................................................................... 10
      2.2.6 Wireshark ........................................................ 11
      2.2.7 DroidWall .......................................................... 11
      2.2.8 Android Debug Bridge ...................................... 11

3 Trace Collection Methodology ...................................... 12
   3.1 Environment Setup .................................................. 12
   3.2 Game Selection ........................................................ 13
   3.3 Trace Collection ....................................................... 14
   3.4 Data Analysis .......................................................... 16

4 Monkey Gamer Evaluation .......................................... 18
   4.1 Percentage of Games Covered by Monkey Gamer ............ 18
   4.2 Monkey Gamer vs. Human Player ............................... 19
   4.3 Summary of the Monkey Gamer Evaluation ..................... 23

5 Profiling Android Games ............................................. 24
   5.1 Description of the Selected Games ............................... 24
   5.2 Communication Energy of Android Games ....................... 27
   5.3 Further Observations ................................................ 31
5.4 Tower Madness vs. Elemental Tower Defense..................................................34
5.5 CPU Utilization.................................................................................................36
5.6 System Call Quantification................................................................................37

6 Conclusions..............................................................................................................40
   6.1 Future Work.......................................................................................................41

Appendix A, Game classification by user input.........................................................45
Appendix B, Links to Google Play Store..................................................................48
Appendix C, System call descriptions.......................................................................53
List of Abbreviations

**Technical Abbreviations**
ADB – Android Debug Bridge  
CSV-file – Comma Separated Values file  
DCH – Dedicated Channel  
FACH – Forward Access Channel  
JSON – Javascript Object Notation  
PCH – Cell Paging Channel  
RNC – Radio Network Controller  
RRC – Radio Resource Control  
UE – User Equipment

**Game Abbreviations**
AB – Angry Birds  
ABGO – Angry Birds GO!  
ABR – Angry Birds Rio  
ABSW – Angry Birds Starwars  
AL – Arcane Legends  
BW – Bubble Witch  
CCS – Candy Crush Saga  
CCSS – Candy Crush Soda Saga  
CTR – Cut the Rope  
EFK – Empire Four Kingdoms  
ETD – Elemental Tower Defense  
RTT – Race the Traffic  
SD – Sonic Dash  
SD3D – School Driving 3D  
SL – Star Legends  
TM – Tower Madness  
QB – Quiz Battle
Chapter 1

Introduction

This report describes the work performed in the context of a Bachelor thesis project at the Real-time Systems Laboratory within the Department of Computer and Information Science at Linköping University. The project was performed by two students in order to complete the Bachelor degree in Computer Engineering.

This chapter introduces the reader to the project. First the context and goals of the thesis are described as well as the planned approach and the limitations of the work. The chapter also presents related scientific work, the structure of the report and the intended audience of the thesis.

1.1 Context and Motivation

With the arrival of smartphones and operating systems like Android and iOS, the popularity and quality of games has increased drastically. With the release of Google Play Store and Apple App Store in 2008 an ecosystem was formed where developers could easily distribute their games and a much wider variety of games could be accessed by the public. In July 2014 Google announced that Android has over 1 billion active monthly users [1] and Google Play Store offers 1.3 million applications [2] where around 26% of all applications are games [3].

Mobile games often connect to the network even though they do not demand always-on network connectivity in order to play. Transmitting data over cellular networks consumes high energy if not done efficiently. The popularity of mobile games and the high energy consumption of the 3G interface causes the phones to consume a lot of energy when games are played. This energy can feel unnecessary to the gamer because it does not always contribute to a good gaming experience and it will significantly reduce the battery lifetime.

Another high energy consuming component in smartphones is the CPU. Previous studies have shown that the CPU is the second highest consumer after the network interface [22]. Research on the CPU-utilization of games is interesting and can reveal potential energy inefficiencies.

The number of malware-infected Android applications in the Google Play Store nearly quadrupled in 2013 compared to 2011 [15], which shows that mobile threats are increasing. Examples of malware applications are clones of popular games (e.g., Flappy Bird [16]). This malware could be used to make calls, install additional apps, send and receive SMS messages, extract contact data, track geo-location, and establish root access, which would allow uninhibited control of the mobile app.
It would be interesting to collect more information about the behaviour of Android games from a security perspective. But a first step towards such studies is a system call profiling step.

Energy consumption and profiling are therefore two important aspects for mobile applications, and therefore for mobile gaming.

1.2 Goals

Profiling mobile games for analysing their CPU energy footprint, communication energy and security aspects can reveal energy inefficiencies and provide interesting empirical data for future studies. This thesis work studies Android games with focus on 3G communication, CPU utilization and system calls.

The goals of the thesis are:

- Profile, analyse and compare the wireless data communication of a selection of Android games. The main subject of the study is the communication energy consumption.
- Analyse and compare the CPU utilization of a selection of Android games.
- Profile and analyse the system calls performed by a selection of Android games. The study aims at quantifying the number and type of calls to be used in later investigations.

1.3 Approach

This section describes the approach to meet the goals described above. In particular, the game selection methodology, the tool selection and the general approach are briefly described.

The game profiling consists of two parts: a resource study and a system call study. The resource study focuses on measuring and analysing the communication energy consumed by the 3G interface while playing Android games, as well as measuring the games' CPU utilization.

The system call study will analyse the kernel-level activity by collecting and analyzing the system calls made. More specifically, the most common system calls will be quantified. The collected system calls can be used to perform security analysis or to model the game behaviour.
In this project we consider using an automatic game profiler, Monkey Gamer [14] for trace collection. Monkey Gamer can play games without user interaction and can therefore potentially save time that would otherwise be spent by humans. First, an evaluation of Monkey Gamer will be performed in order to determine if it is a suitable tool for trace collection in the context of an energy study.

The project is divided into four main phases:

1. **Monkey Gamer evaluation:**
   One of the main tasks of this project is to collect traces. Monkey gamer can automatically play games and collect traces with minimum user interactions. The project aims at collecting realistic traces, and therefore an evaluation will be performed to evaluate whether the energy consumed by the phone when the games are played is the same for Monkey Gamer and human players. Currently, Monkey Gamer cannot play all types of games, and thus the evaluation will analyse to which extent this can become a limitation by analysing the top 100 free games of the Google Play Store.

2. **Game selection:**
   The list of selected games should include many different kinds of games, that are comparable due to common but also different properties. To find the right games the Google Play Store will be examined for popular and interesting games that fit the analysis. The strategy for picking the games is further explained in Chapter 3.

3. **Trace collection:**
   To perform the trace collection, a smartphone will be employed. This will be connected to a Linux computer via USB. The traces that will be collected are traces of the data packets sent and received by the games, system calls traces and CPU utilization traces. Collecting these traces will be done using appropriate Unix-command line tools. The tools run on the phone collecting information while the games are being played.

4. **Data analysis:**
   The analysis will compare the games and draw conclusions about the energy consumption, network usage and system calls of the selected games. The analysis of the collected data will be performed using tools such as Wireshark and EnergyBox [11], as well as custom scripts. The usage of the tools are explained in Chapter 3.
1.4 Limitations
The selection of games is limited to the top 100 free games of the Swedish version of the Google Play Store and this is reflected in the game selection of this project. Aspects like GPU utilization are out of the scope of the study.

1.5 Intended Audience
This thesis would be of interest to someone who studies computer science and is interested in energy studies. The thesis could also be of value to someone who develops Android games and wants to know if communication energy in Android games is something to consider. It is also partially accessible for anyone who is interested in Android games and is concerned about the battery lifetime of their smartphone.

The background chapter introduces the needed concepts to understand the rest of this thesis.

1.6 Related Scientific Work
This section describes a selection of works related to this thesis, where the authors also studied energy and profiling on smartphones.

1.6.1 Energy-Related Studies
Prokova et al. [4] study how much energy a smartphone consumes due to the data communication needed to fetch in-game advertisements in Android games. The authors measured the energy consumption by comparing the average power usage of Android games with and without advertisements while the phone was connected to a WiFi network. The authors found that a high frequency of ad-requests consumes more energy. This thesis does not focus on advertisements but the total power consumed due to data traffic related to games when connected to a 3G network.

Pathak et al. [5] present eprof, a tool for developers to analyse the energy consumed by applications. The work covers a case study where eprof is used to profile popular apps, including Angry Birds. According to them only 25-35 % of the energy is spent on actual gameplay and the rest is spent on user tracking and networking activities. This thesis extends their work by covering the energy consumed by a larger set of Android games showing the variety in energy consumption.
Metri et al. [19] study how the energy consumption of smartphones varies depending on background applications and network connection type. They show that the energy efficiency of an iPhone can be increased by 59% when streaming music on WiFi instead of 3G. They also show that network applications running in the background, when using 3G, can reduce the energy efficiency of an iPhone by up to 79% compared to staying in the idle state.

Vergara et al. [8] present an energy study on mobile instant messaging applications. The paper shows that even simple short messaging sessions over cellular networks using instant messaging applications consume a lot of energy. The energy consumption of instant messaging applications presented in the paper can give us a greater understanding of how mobile games compare to other mobile activities. The energy consumed by instant messaging applications is measured using EnergyBox [11] which is considered as a tool in this thesis for estimating the communication energy of Android games.

Ma et al. [13] present an in-depth quantitative bottleneck analysis on performance and power consumption among games' 3D graphics pipeline. They selectively disabled one or more graphics pipeline stages to identify and quantify the performance and power consumption bottlenecks. This thesis is similar because it also studies energy in mobile games but focuses on the energy consumption related to data communications in games instead of 3D graphics. Moreover this thesis focuses on both 2D and 3D games.

1.6.2 Profiling-Related Studies

Wei et al. [6] present Profiledroid, a multi-layered Android-application for monitoring and profiling Android applications. The profiler analyses four layers: application specification, user interactions, operating system and network traffic. The authors proposed an ensemble of metrics at each layer to capture the essential characteristics of app specification, user activities, OS and network statistics. The authors focus on the system call intensity and data traffic intensity of general Android applications whereas this thesis focuses on the data traffic and system call intensity of Android games.

Ham and Lee [9] present a security study that focuses on system calls made by benign and malware applications and showing differences in their system call behaviour. They found that the system calls bind, brk, connect, fdatasync, mkdir, msgget, pwrite, recv, recvfrom, rename, rt_sigreturn, semget, semop, setsockopt, socket, statfs64, SYS_224 and SYS_248 were only found in malicious applications. This thesis compares the system calls made by games with the system calls made by benign and malware applications presented by the authors to show how games differ from other applications.
Burguera et al. [20] present Crowdroid, a framework to obtain and analyse smartphone application activity. The framework can accurately distinguish between benign and malware versions of applications based on their system call behaviour. The authors indicate that monitoring system calls is a potential method for detecting malware.

1.7 Report Outline

The structure of this report is as follows: Chapter 2 explains the background of this project by explaining concepts and tools that are needed to understand other parts of the report. Chapter 3 describes the methodology followed to complete the various tasks faced in this project. Chapter 4 presents the evaluation of Monkey Gamer. In Chapter 5, the results of the energy and system call studies are presented as well as observations made on the subject. Chapter 6 concludes the work presented in this report and proposes future work related to the thesis.
Chapter 2

Background

The background chapter introduces the needed background to understand the rest of the thesis. The energy consumption due to 3G communication and the different tools used in the thesis are described.

2.1 3G Energy Consumption

The energy consumed of the user equipment (UE), like smartphones or tablet computers, connected to a 3G network is determined by the radio resource management performed at the network operators side by the Radio Network Controller (RNC) [12]. The RNC uses the Radio Resource Control protocol (RRC) to handle the control plane signaling between the UE and the network operator. According to the RRC protocol the UE can be in different states where each state has a different power consumption and performance in terms of data rate and latency. The state of the UE is decided by the RNC.

As Figure 2.1 describes, there are 3 different RRC states: PCH (Cell Paging channel), FACH (Forward access channel) and DCH (Dedicated channel). There is also an idle state for UEs that are not currently connected to the network operator. Transitions to higher states are determined by the amount of data that needs to be sent and transitions to lower states are controlled by inactivity timers.

The inactivity timer specifies the amount of time the UE will remain in the current state when it has finished sending/receiving data. The inactivity timers prevent the UE from transitioning to a lower state when sending data within short periods of time thus avoiding the delay of state transitioning.

Figure 2.1. Illustration of the RRC state machine. Adapted from [11].
In the DCH state the UE is allocated a dedicated channel providing the highest data rate. This is the state with the highest performance but also the highest energy consumption. The UE transitions to FACH when the inactivity timer T1 is expired, i.e., when there is no data sent for T1.

In the FACH state the UE can send and receive data at a much lower data rate compared to DCH but also at a lower energy cost. The UE transitions to the DCH state when the data that needs to be transmitted exceeds a certain threshold and it returns to the PCH-state when the inactivity timer T2 is expired.

In the PCH state, the UE is idle but is connected to the network operator. No channel is allocated and no data can be sent or received but the UE can quickly transition to FACH or DCH when data is ready to be transmitted. The UE transitions to the idle state and is no longer connected when the inactivity timer T3 is expired.

When it comes to energy consumption, the inactivity timers can cause problems. The amount of time the UE remains in the current state due to inactivity timers can lead to a waste of energy if the UE did not mean to send any more data during the inactivity timers as seen in Figure 2.2.

In Figure 2.2, data is sent during a short period. The green parts of the figure shows the data transfer which occurs in the DCH state. The blue parts shows where energy is spent in the higher power states during inactivity timers where no data is transferred. We call this energy an "energy tail". In the case shown in the figure, energy would have been saved if there were no inactivity timers and if the UE instead transitioned to the idle state when it was done transmitting. Since the transition from PCH to a state where the UE can send data is long (e.g., 1 second), the inactivity timers reduce the delay.

![Figure 2.2. Illustration of the "energy tail" generated by the inactivity timers. Adapted from [10].](image-url)
2.2 Description of Used Tools

This section presents an overview of the tools that were used in this thesis.

2.2.1 Monkey Gamer

Monkey gamer [14] is a tool that can automatically play Android games and collect execution traces containing information such as system calls and network traffic. It was created by Javier Marian Santos as part of a Master Thesis project at Linköping University in 2013 [14, 21]. The tool runs on a computer and then installs, launches and plays the games on a smartphone connected via USB. Its job is to explore the content of the game and can therefore effectively replace a human tester.

Manual profiling and testing demands human time. The advantage of using Monkey Gamer is that playing and testing an Android game requires minimal effort from humans as it is all done automatically. Using Monkey Gamer can therefore reduce the cost of testing Android games. However, not all games can be played by Monkey Gamer because it uses an object-detection system that needs time to find and click on objects that it can interact with. This creates a limitation because it can only play games with a certain input type. This limitation as well as the input types are analysed in Chapter 4, where the evaluation of Monkey Gamer is presented to determine if the tool is suitable for collecting traces for this project.

2.2.2 EnergyBox

EnergyBox [11] is a tool that accurately estimates the 3G and WiFi energy consumption of a UE. EnergyBox takes packet traces as its input and outputs information such as the energy consumption and RRC state transitions. EnergyBox uses the packet traces together with network configuration parameters and outputs the UE’s RRC states over time. It then uses these states together with device specific parameters to estimate the consumed energy. The configuration parameters required by EnergyBox are inactivity timers, transition delays, buffer thresholds and specific power levels of each RRC state for the specific UE that is simulated. EnergyBox has been shown to be 98% accurate for both 3G and WiFi using fairly large evaluation data.

The advantage of EnergyBox is that you do not have to physically measure and calculate the energy consumed by hardware but instead let software record the network traffic and run EnergyBox with the collected traces.

In our project, EnergyBox is used together with the collected packet traces in order to estimate the communication energy consumed by the games that were played. The state occupancy and state transition information is used to gain more details about the 3G transmissions.
2.2.3 Tcpdump

Tcpdump\(^1\) is a Unix command line tool that allows the user to monitor network traffic and capture or dump it to a file. The tool captures packets being transmitted or received by the host running tcpdump.

Tcpdump is used in this project to produce packet traces containing information about the packets transmitted and received by the games. The packet traces are used as inputs to EnergyBox.

2.2.4 Strace

Strace\(^2\) is a debugging tool in Linux that can be used to monitor system calls and signals made by processes. It can be configured for a specific process to record all system calls made, how many times they were called and how much time was spent in each.

In this project Strace is used to collect information about system calls made by the games. Its ability to capture how many times each system call is made is particularly useful.

2.2.5 Top

Top\(^3\) is a Unix command line tool that continuously produces a list of processes running on the system and displays CPU usage, memory usage and runtime for each process. It is used for detecting how much system resources the users and processes are consuming. The tool can also be configured to only display information related to a specific process.

For this project, the interesting part of this tool is its ability to monitor the CPU utilization of running processes. Top is used to monitor and record the CPU utilization of the games played.

---

2. http://linux.die.net/man/1/strace
3. http://linux.die.net/man/1/top
2.2.6 Wireshark

Wireshark\(^1\) is an open-source packet-analyser that, like Tcpdump, can be used to capture packets sent and received over the network. The tool can also be used to analyse captured traces as it shows valuable information about packets such as size, round-trip-times, flags, content, protocols used etc. Unlike Tcpdump, Wireshark has a graphical user interface and comes with sorting and filtering capabilities.

Because of these properties, Wireshark is used in this project for analysing the traces collected by tcpdump to gain more information about the traffic produced by the games.

2.2.7 DroidWall

DroidWall\(^2\) is a software firewall that allows the user to restrict which applications can access the network. This can be used for security reasons or to make the battery last longer. To run DroidWall you need root access on your device. Droidwall is based on the iptables Linux Firewall.

DroidWall is used in this project to isolate network traffic of the game under test, so that Tcpdump only captures packets related to the games to collect only relevant traces.

2.2.8 Android Debug Bridge

Android Debug Bridge (ADB)\(^3\) is a command line tool that allows the execution of Linux commands on a smartphone from a computer. ADB is used to start Top, Tcpdump and Strace on the phone to collect traces while the games are being played. ADB is also used by Monkey Gamer to install the games using android application package files.

\(^1\) [https://www.wireshark.org/](https://www.wireshark.org/)
Chapter 3

Trace Collection Methodology

This section describes the methodology followed to select the games to profile, to collect traces and to extract data for the analysis.

3.1 Environment Setup

The hardware used in the project is a laptop running Ubuntu version 12.04, a Samsung Galaxy S3 smartphone running Android version 4.1.2 and a USB cable to connect the devices. All the games were played in the Samsung Galaxy S3 smartphone.

The tools used in the laptop are the following:

- Android Debug Bridge
- Wireshark
- EnergyBox
- Monkey Gamer

The following tools are used in the smartphone:

- Tcpdump
- Top
- Strace
- DroidWall

Figure 3.1 illustrates the hardware setup of the experiments. The laptop is used to store the collected data, to start relevant tools on the phone by using Android Debug Bridge, as well as analyzing packet traces with the help of Wireshark and EnergyBox. The smartphone was "rooted" because it is a necessity for certain tools to work (e.g., Tcpdump, Droidwall, Strace, Monkey Gamer).
3.2 Game Selection

An important part of this project is the selection of games that are going to be profiled. The selected games should be representative of other games on the market. Some of the chosen games should be of the same kind so that they can be compared. Moreover, other games should be quite different to analyse different patterns. Our game selection strategy is meant to allow us to characterize communication patterns based on the properties of the games. The properties considered were:

- **Gameplay and objectives**
  Some games should have different kinds of gameplay and objectives and some games should have similar gameplay so they can be compared with each other to draw conclusions.

- **Single player or online game**
  An online game is expected to produce a lot more network traffic since it is constantly connected to communicate with other players and servers. Comparing single player with online games will show different communication patterns.
• **Release date**
  The games selected should consist of older and newer games to try and identify possible trends, e.g. if there is a larger problem in newer games and if the situation could get worse in future games.

• **Input type.**
  The inputs we considered are the time-sensitive and time-insensitive input types which are described in Section 4.1.

We have some expertise in mobile games as users, which was used to select the interesting candidates. Each game was selected on the following basis:

• Properties (input type, release date, single player or online game)
• Similarity to the other selected games.
• Authors' previous experience.

The final list of selected games is presented in Chapter 5.

### 3.3 Trace Collection

The trace collection is an important part of this project because data to analyse is needed to be able to say something about the data transmissions of Android games and to propose changes that may improve the energy consumption. The traces are basically text-files containing run-time information about a process; in our case the mobile games that are played. The information that is collected while the games are being played are the system calls with Strace, CPU utilization with the command Top and the network traffic by using Tcpdump.

ADB is used to simplify the process of starting the tools on the smartphone. ADB allows the needed tools to be started on the phone from the Linux computer. The alternative is to start the tools directly on the phone using an emulated Unix terminal and the phones limited interface but using ADB is a more efficient method.
Figure 3.2 shows the process of starting the trace collection tools. The process is described next:

1. ADB is launched on the laptop and enables to run the commands on the phone. DroidWall was enabled on the phone to isolate the traffic to the game under test.
2. Tcpdump is launched on the phone using ADB on the laptop. Tcpdump is launched before starting the game to ensure that all the traffic that is sent when the game is initializing and connecting to various servers is collected.
3. The game is launched on the phone.
4. The process id is fetched from the laptop using the ps-command which is a unix command that lists information about all the running processes on the system.
5. Top and Strace is launched using ADB. The process id is used to attach Top and Strace to the games process so they only gather information related to the game.
6. The game is played on the phone.

The relevant tools are now running on the phone and are actively collecting information about the game that is being played.

Figure 3.2. Illustration of the trace collection process.

15
Playing the same game twice will not produce the exact same traces and the results can vary. Therefore, the following experimental methodology was employed. Each game is played 5 times and the duration of each run is exactly 15 minutes. The 5 runs is then used to calculate averages that better represents a typical playing session. According to our experiences, playing 15 minutes is enough to replicate a normal gaming behaviour and the data collected from the traces should be able to represent the real life situation. A gaming sessions is also quite repetitive so playing longer sessions does not necessarily add more information.

3.4 Data Analysis

Once the traces are collected, the data analysis attempts to extract interesting data from them. The metrics used to analyse the collected data are the communication energy consumption, the amount of data transmitted, the RRC state time distribution and the number of RRC state transitions made by the selected games. This information is not directly readable from the packet traces but can be extracted from them by using suitable tools and scripts.

Wireshark is used to obtain information about the packet traces collected with Tcpdump. Wireshark's GUI is used to analyse the packet traces, to gain an understanding of the networking behaviour of each game and to determine what the data connections are used for. Another function of Wireshark is to print out a summary of the packets collected, which is used to gather the amount of data transmitted by each game.

EnergyBox is used to extract valuable information and to study the energy consumption of games. Given a packet trace, EnergyBox outputs the 3G energy consumption and RRC state transition information in form of illustrative graphs and detailed Comma-separated values (CSV) files [17] which is used for the analysis. The CSV files containing the RRC state transition information was used as input in a script we created in order to calculate the amount of time the UE spends in each RRC state and the amount of state transitions that were made. The time spent in each RRC state is interesting because it shows the time the games spend in the energy expensive DCH and FACH states as well as the lower consuming PCH state. This shows how the energy is spent and if the games behave efficiently. The amount of RRC state transitions is interesting because it can reveal more information about the data patterns of the games.

The configuration parameters of EnergyBox are set as seen in the Tables 3.1-3.4 below and are obtained from the EnergyBox evaluation settings [11]. The parameters correspond to the Sweden 3G network operated by TeliaSonera measured at Linköping University in 2012.
The RLC buffer thresholds are the following:

<table>
<thead>
<tr>
<th>State Transition</th>
<th>Downlink size (Bytes)</th>
<th>Uplink size (Bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCH-DCH</td>
<td>515</td>
<td>850-1000</td>
</tr>
<tr>
<td>FACH-DCH</td>
<td>515</td>
<td>294</td>
</tr>
<tr>
<td>PCH-FACH</td>
<td></td>
<td>Is always triggered</td>
</tr>
</tbody>
</table>

*Table 3.1. RLC buffer thresholds.*

The state transitions are modelled as a constant time in EnergyBox, which are shown in Table 3.2.

<table>
<thead>
<tr>
<th>State Transition</th>
<th>Transition Delay (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCH-DCH</td>
<td>1700</td>
</tr>
<tr>
<td>FACH-DCH</td>
<td>650</td>
</tr>
<tr>
<td>PCH-FACH</td>
<td>435</td>
</tr>
</tbody>
</table>

*Table 3.2. Transition delays.*

The inactivity timers of the modelled network are:

<table>
<thead>
<tr>
<th>State inactivity timer</th>
<th>Timer value (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1 (DCH-FACH)</td>
<td>4100</td>
</tr>
<tr>
<td>T2 (FACH-PCH)</td>
<td>5671</td>
</tr>
</tbody>
</table>

*Table 3.3. Inactivity timers.*

EnergyBox also requires the device-specific power values to model the consumption of the device. These are obtained from the measurements performed in a related article [12].

The power of the PCH state is set to 0 in order to measure only the communication consumption, since the measurement period (15 minutes) is the same for all the games.

<table>
<thead>
<tr>
<th>State</th>
<th>Power consumption (W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCH</td>
<td>0</td>
</tr>
<tr>
<td>FACH</td>
<td>0.400</td>
</tr>
<tr>
<td>DCH</td>
<td>0.612</td>
</tr>
</tbody>
</table>

*Table 3.4. Device specific power consumption.*
Chapter 4

Monkey Gamer Evaluation

The purpose of using the automatic game profiler Monkey Gamer is to automatize the collection of traces. Monkey Gamer can be potentially used in order to reduce the time spent in performing manual testing. However, a part of this thesis is meant to show how data traffic in mobile games effects the battery life time of the player's smartphone. The aim of this chapter is to evaluate the usefulness of Monkey Gamer for this purpose.

The main aspects that we evaluate are:

- How can the selection of games be limited by the fact that Monkey Gamer can only play certain type of games?
- Is the communication energy consumption of a smartphone the same when Monkey Gamer and a human plays the games?

4.1 Percentage of Games Covered by Monkey Gamer

As mentioned in section 2.2.1, Monkey Gamer can only play games with a certain input type. The games that it can play require the time-insensitive input type. Santos et al [14] define these input types in the context of different types of games as:

- Time-insensitive: the input of the game determines the state of the game. The state of the game changes only when an input is given by the user. For example, Tic-tac-toe.
- Time-sensitive: the input type of games where the state can change without any user input. For example, driving games where the user has to turn at the correct time, or games where the player controls a character's movement and has to avoid obstacles such as pitfalls or enemies.
- Intelligence: The inputs of these games are given in text format. For example, quiz games.

The method for evaluating the number of games that were playable by Monkey Gamer is as follows: The top 100 free games from the Google Play store were chosen at the 26th of May, 2014. Each game is studied and put into one of the following categories: Time-sensitive, Time-insensitive or Intelligence game.
This analysis provides us with an idea of how big the limitation of Monkey Gamer is and which popular and interesting games cannot be used in this study.

Figure 4.1. Classification of the top 100 free games of the Google Play Store by user input.

Figure 4.1 shows the result of the game classification by user input. The result shows that 59 % of the games are time-sensitive, 37 % are time-insensitive and 4 % require human intelligence. This tells us that approximately 37% of the games on the market can be played by Monkey Gamer. Note that the result is an approximation since some of the games require more than one type of input. A comparison with the same study performed in 2013 shows that the number of games that can be played by Monkey Gamer is similar (45% in 2013) [14]. However, since the 100 games change over time, the results vary over time.

4.2 Monkey Gamer vs. Human Player

The method to evaluate whether Monkey Gamer can be used to create the inputs in the trace collection and the results are described in this section. The methodology used is to compare the resulting energy consumption when a human and Monkey Gamer plays the games. We argue that a similar energy consumption would reflect the fact that Monkey Gamer can be used in order to collect realistic traces.

The experiment is performed by using the trace collection methodology described in Chapter 3 for three different games. The games that are used in this comparison are Angry Birds, Cut the Rope and Candy Crush Soda Saga. Monkey Gamer and a human player plays each game 5 times, 15 minutes each run. The traces are used as inputs to EnergyBox to estimate the average energy consumption which is then compared.
Figure 4.2 shows the average energy consumption of the three games when played by Monkey Gamer and the human player for the same amount of time.

![Energy Consumption Graph](image)

Figure 4.2. 3G communication energy consumed by the games when played by Monkey Gamer and a human player.

The Figure shows that the games consumed a lot more energy when the human played compared to Monkey Gamer and that the difference varies from game to game. For Cut the Rope and Candy Crush Saga the communication energy consumed by Monkey Gamer is close to half of what a human consumed but for Angry Birds the difference is much larger.

The difference in energy consumption can be explained by the fact that Monkey Gamer explores the games a lot slower than a human and therefore reaches parts of the games where data is transmitted, less frequently. This means that in a 15 minute comparison there will be fewer data transmissions when Monkey Gamer plays and therefore it consumes less energy.

The reason that Monkey Gamer is slower than a human is that it needs time to analyse screens, detect interactive objects and determine the next action. On average it takes Monkey Gamer about 15-20 seconds to perform an action. We have also observed that Monkey Gamer spends a significant amount of time in certain screens that a human would normally pass in only a few seconds. An example is the main menu of Candy Crush Soda Saga, that a human pass in about 2-5 seconds and Monkey Gamer pass in approximately 10 minutes.
The reason that Monkey Gamer spends a lot of time in certain screens is that its object-detection system detects false objects that are not actually interactive. Monkey Gamer provides object-detection screenshots that clearly show where it has detected objects that it believes are interactive. Figure 4.3 and 4.4 are examples of object-detection screenshots produced by Monkey Gamer. The blue dots in the figures shows where Monkey Gamer has detected a clickable object.

In Figure 4.3 the objects that it has detected that are interactive are the ones on the green Play-button and the blue Connect-button. It has also detected a number of objects outside of these buttons that are not interactive. What makes Monkey Gamer spend so much time in this screen is the fact that it has detected 34 objects where only four of them are clickable. It will interact with all of these objects and spend time analyzing what happens when each object is interacted with. The time that it spends interacting with false objects is wasted and slows down the exploration of the game.

Figure 4.4 is also an interesting example of an object-detection screenshot produced by Monkey Gamer. This is a screen that Monkey Gamer could not pass through. The figure shows that Monkey Gamer has not detected any of the objects that are clickable and will lead to new screens.

Improving the object-detection system should improve the speed of Monkey Gamer to make it closer to a human player.
Figure 4.3. Erroneous object detection by Monkey Gamer. 30 out of 34 objects are not clickable.

Figure 4.4. Erroneous object detection by Monkey Gamer. No clickable object is found.
4.3 Summary of the Monkey Gamer Evaluation

The classification by user input of the top 100 free games on the Google Play Store shows how Monkey Gamer would limit what games can be used in this study. We think that only being able to play 37% of the games would limit the results of the study since more than half of the games are eliminated from our game selection. Popular games like Angry Birds GO! And Sonic Dash as well as online multiplayer games that could be interesting to compare with single player games can not be played by Monkey Gamer.

We have seen that Monkey Gamer explores the game slower than a human because it needs time to decide which action to take and analyse where each action leads. The slow tempo, together with the fact that it spends too much time in certain screens will affect the RRC state behaviour of the UE when playing games. Since the energy consumption of user equipments that are connected to a 3G network is determined by the RRC logic, the frequency in which the player performs actions matter. This means that even if Monkey Gamer performs the same tasks as a human the energy can differ. This can make the energy consumption of the traces collected by Monkey Gamer inaccurate and they cannot be used instead of human traces in this context.

However, Monkey Gamer is a useful tool for other purposes such as general game testing, finding bugs and unexpected behaviour in mobile games and could effectively replace a human tester in that context. Therefore, for the purpose of this project, the games are played by human players leading to a more interesting game selection and completely human-like traces.
Chapter 5

Profiling Android Games

In this chapter the results of the energy and system call studies are presented. The chapter begins with a short description of the games that were selected. Then the measurements and the collected data as well as observations made from the energy study. A detailed comparison between two of the selected games is also presented. Finally the CPU utilization and system call analysis is presented.

5.1 Description of the Selected Games

In this section we give a brief description of each of the 20 games that were selected for this study. The time frame of this project limited the amount of selected games to 20. In particular, the focus of the description is on the gameplay, input type, if they use in-game advertising and the version of the games that were used.

The description of the selected games is shown in Tables 5.1-5.3 based on the categories. The main categories are: single player games with the time-insensitive input type, single player games with the time-sensitive input type and online games. Appendix B contains screenshots and links to each games' Google Play Store page.

Single player, Time-Insensitive inputs

<table>
<thead>
<tr>
<th>Game</th>
<th>Gameplay</th>
<th>Advertisements</th>
<th>Release Date</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angry Birds (AB)</td>
<td>A 2D game where birds are fired at enemies from a slingshot.</td>
<td>Yes</td>
<td>December 2009</td>
<td>2.2.0</td>
</tr>
<tr>
<td>Angry Birds RIO (ABR)</td>
<td>A newer version of Angry Birds with the same kind of gameplay and objectives.</td>
<td>Yes</td>
<td>March 2011</td>
<td>2.2.0</td>
</tr>
<tr>
<td>Angry Birds Starwars 1 (SW1)</td>
<td>Another version of Angry Birds with a new theme.</td>
<td>Yes</td>
<td>November 2012</td>
<td>1.5.2</td>
</tr>
<tr>
<td>Angry Birds Starwars 2 (SW2)</td>
<td>A newer version of Angry Birds Starwars 1 with more advanced graphics and similar gameplay.</td>
<td>Yes</td>
<td>September 2013</td>
<td>1.6.0</td>
</tr>
<tr>
<td>Cut the Rope 1 (CTR1)</td>
<td>2D physics-based puzzle game. The player cuts ropes to feed candy to a monster.</td>
<td>Yes</td>
<td>October 2010</td>
<td>2.4.2</td>
</tr>
</tbody>
</table>
### Single player, Time-Sensitive inputs

<table>
<thead>
<tr>
<th>Game</th>
<th>Gameplay</th>
<th>Advertisements</th>
<th>Release Date</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angry Birds GO!</td>
<td>3D kart-racing game where the player controls an angry bird through a series of tracks.</td>
<td>Yes</td>
<td>December 2013</td>
<td>1.4.3</td>
</tr>
<tr>
<td>Sonic Dash</td>
<td>3D endless runner platform game where the player directs Sonic through levels, collecting rings and avoiding obstacles and enemies.</td>
<td>Yes</td>
<td>March 2013</td>
<td>1.14.0.Go</td>
</tr>
<tr>
<td>Race the Traffic</td>
<td>3D game where the objective is to drive a car and avoid traffic.</td>
<td>Yes</td>
<td>July 2014</td>
<td>1.0.6</td>
</tr>
<tr>
<td>School Driving 3D</td>
<td>A 3D game where the player controls a car through a city learning how to drive.</td>
<td>Yes</td>
<td>April 2014</td>
<td>1.7.0</td>
</tr>
<tr>
<td>Elemental Tower Defense</td>
<td>A tower defense game where the player builds towers that shoot at enemies trying to reach a certain point on the map.</td>
<td>Yes</td>
<td>May 2014</td>
<td>1.0.57</td>
</tr>
</tbody>
</table>

### Table 5.1. Description of the single player, time-insensitive games.
Tower Madness (TM)   Another tower defense game.   Yes   October 2009   1.21

*Table 5.2. Description of the single player, time-sensitive games.*

## Online Games

<table>
<thead>
<tr>
<th>Game</th>
<th>Gameplay</th>
<th>Advertisement</th>
<th>Release Date</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Star Legends (SL)</td>
<td>Multiplayer Sci-fi Role-playing game, fighting monsters with other players over the internet.</td>
<td>No</td>
<td>Q2 2011</td>
<td>2.0.1.0</td>
</tr>
<tr>
<td>Arcane Legends (AL)</td>
<td>Multiplayer fantasy Role-playing game.</td>
<td>No</td>
<td>November 2012</td>
<td>1.1.0.1</td>
</tr>
<tr>
<td>Quiz Battle (QB)</td>
<td>Turn-based quiz game, to play against opponents over the internet.</td>
<td>No</td>
<td>August 2012</td>
<td>1.3.102</td>
</tr>
<tr>
<td>Empire Four Kingdoms (EFK)</td>
<td>Build a medieval castle, create an army and fight player versus player battles over the internet.</td>
<td>No</td>
<td>March 2013</td>
<td>1.10.60</td>
</tr>
</tbody>
</table>

*Table 5.3. Description of the online games.*
5.2 Communication Energy of Android Games

This section presents the results from the energy study. The communication energy consumption due to 3G communication, the amount of data sent, the RRC state time distribution and the amount of RRC state transitions of the selected games are presented.

Figure 5.1 shows the average communication energy consumption as well as the standard deviation of the five runs of each of the 20 games selected. The green color represents time-insensitive games, the blue color represents time-sensitive games and orange represents online games.

Figure 5.1 shows that there is a big difference in the energy consumption of the 20 games. The highest consuming game selected is Arcane Legends (AL) with a mean energy consumption of 561 Joules which is approximately 2% of a smartphone's battery. The least consuming among the selected games is Tower Madness (TM) with a mean energy consumption of 65 Joules. This is evidently a big diversity and shows that different games can effect the battery lifetime of the phone unequally. Of the 20 selected games, half of them consume more than 435 Joules. This shows that a high percentage of the games consume a lot of energy.

Arcane Legends (AL), Star Legends (SL) and QuizBattle (QB) are among the highest consumers. This result is expected since they are online games and communication over the network is required so that the gaming session is allowed to be synchronized for the involving players. Surprisingly, single player games like the Angry Birds games (AB, ABR, SW2, ABGO) and the Cut the Rope games (CTR1, CTR2) that do...
not require a network connection still consume almost the same as some online games.

Figure 5.2 shows the average amount of data transferred and standard deviation of the five runs of each of the 20 games we selected. The average energy consumption of each game can again be seen below each bar in the figure.

The figure shows that there is a huge diversity in the amount of data transferred by the 20 games. Moreover the standard deviation for some games is large which means that the amount of data sent differs a lot for each run. The largest amount of data was transferred by Angry Birds RIO (ABR) and the least amount of data was transferred by Bubble Witch Saga 1 (BW1) with 13664 KB and 66 KB respectively. The amount of data varies because the number of ads fetched and statistics sent is different for each game and the ads comes in different forms such as video or images.

By analyzing Figure 5.2 it becomes clear the total energy consumed is not proportional to the amount of data transferred by the games. A very clear example is given when comparing Elemental Tower Defense (ETD) and Tower Madness (TM). Approximately 63 times more data was transferred by Tower Madness (4576 KB compared to 73 KB) but it consumed about seven times less energy than Elemental Tower Defense (65 J compared to 444 J). This is a clear example that the communication energy consumed is not really decided by the amount of data sent. A detailed comparison between Tower Madness and Elemental Tower Defense can be found in section 5.4.

Figure 5.2. The average amount of data transferred by the 20 games.
The difference in energy and amount of data sent by the single player games with in-game advertisements and without is enormous. Out of the 16 single player games, only Candy Crush Saga (CCS), Candy Crush Soda Saga (CCSS), Bubble Witch Saga 1 (BW1) and Bubble Witch Saga 2 (BW2) which are developed by the same company are ad-free and do not use in-game advertisements. Only these games have a low amount of data transferred together with a low energy consumption. Section 5.3 further discusses what the communication in ad-free games is used for.

In order to analyse how the data pattern of the different games impacts the energy consumption, we analyse the time spent in the different RRC states for each game.

Figure 5.3 shows the RRC state time distribution of the trace with the median energy consumption for each game.

Figure 5.3. The percentage of time the smartphone spends in each RRC state for each game.

To consume as little energy as possible, a game should spend as much time in the lower consuming PCH state as possible and spend as little time as possible in the higher consuming FACH and DCH states.

Figure 5.3 shows that the RRC state time distribution differ a lot and the games seem to have a very different communication behaviour. The highest consuming games spend most time in the most energy expensive DCH state. Angry Birds Rio (ABR), Star Legends (SL) and Arcane Legends (AL), spend approximately 95% of the time in DCH and a negligible amount of time in PCH.

Some games with a very similar kind of gameplay and objectives have a very different RRC state distribution. Some examples are Angry Birds Starwars 1 (SW1) compared to Angry Birds Starwars 2 (SW2) and Angry Birds RIO (ABR), as well as Sonic Dash (SD) and School Driving 3D (SD3D) compared to Angry Birds Go (ABGO). This finding indicates that the networking behaviour of these games can be improved in order to save battery lifetime.
One game that stands out is the online game Empire Four Kingdoms (EFK) which spends the majority of the time in PCH and especially FACH state and seems to almost completely avoid the DCH state. The reason for this is that it does send packets often but the packets are small enough to not overflow the DCH buffer threshold and therefore the UE stays in the less expensive FACH state.

Another interesting observation is that of the single player games that uses in-game advertisements, the majority of the games with the time-sensitive input type (TM, SD, RTT, SD3D) spend more time in PCH than those with the time-insensitive input type (AB, ABR, SW2, CTR1, CTR2). This can be seen by analysing the percentage of the bars that are covered by green for these games in Figure 5.3. When analysing the packet traces, we have noticed that the way the ads are fetched and used seems to play a big role in the RRC state time distribution of the games. The majority of the games fetch ads when entering, restarting or completing levels without any exceptions. In the time-insensitive games we selected, the levels are completed faster than in the time-sensitive games. Approximately 40 seconds is spent completing a level in time-insensitive games which means that advertisements are fetched frequently. In the time-sensitive games, the player spends as much as 2-10 minutes to complete a level and advertisements are therefore fetched with a lower frequency. The frequency of ad-fetches is affected by the amount of time spent completing levels in the games and this could be the reason for the difference in RRC state time distribution of some time-sensitive and time-insensitive games.

Figure 5.4 shows the volume of RRC state transitions that were made by each game from the collected traces. The run with the median energy consumption was chosen for each game. This information can provide insight about the games data patterns.

![Figure 5.4. Amount of RRC state transitions made by the 20 games.](image)
When the UE needs to transition to a new state, signaling is required between it and the 3G operator. The different state transitions result in different amounts of signaling messages. The signaling needed to switch to a new state requires bandwidth and takes time, delaying user packets. So this information is indeed interesting for the mobile operator and system developers.

Like the previous data, there is also a big diversity in the amount of RRC-state transitions made by the games. The total number of state transitions range between 4 and 144.

There is a big difference in the amount of state transitions made by the online games as Start Legends (SL) and Arcane Legends (AL) transitions between states less often than Quiz Battle (QB) and Empire Four Kingdoms (EFK). This can be explained by the fact that SL and AL are real-time games and the world needs to be updated often, resulting in more frequent communications with the game server.

Sonic Dash (SD), Race The Traffic (RTT) and Empire Four Kingdoms (EFK) are the only games where the majority of the state transitions are between the PCH and FACH states. These games seem to use small data transmissions to avoid transitions to the DCH state, which is a good behavior because it allows them to send data at a lower energy cost.

5.3 Further Observations

The transmission energy consumed by Android games is relatively high when using 3G. Also, the energy consumption of games vary a lot which indicates that some games are not optimized and have room for improvements.

**Game statistics:** A deeper analysis of the individual packet traces reveals that the most of the transmissions are used for fetching ads, sending game statistics and encrypted data. The game statistics sent by the games are usually related to the players gaming behavior. More specifically, games send information such as player achievements, score earned, items purchased and money spent. Figure 5.5 shows statistics sent by Bubble Witch 2. The information was obtained from the packet trace using Wireshark. The figure shows the game sends information such as missed shots, maximum consecutive successful shots and stars collected.
Potential optimisations: The energy consumption of Android games could be improved if the developers optimized the data transmissions. Instead of fetching an advertisement everytime the player completes a level, advertisements could be prefetched in bundles and then presented individually at a later stage. This would mean that ads are not fetched as frequently. A similar method could be used for sending level-related statistics. Instead of sending the information as soon as the level is completed, it could be stored and the information of several levels could be sent at the same time. If these methods are used, data would not be sent as often and the UE could remain in the PCH state for a longer period of time.

Blocking the traffic: A method that can be used to avoid the communication energy consumed by the games entirely is to activate a firewall that blocks network traffic from and to the games. All of the 16 single player games we selected can be played and no packets are transmitted when the firewall is activated. Only the online games were not playable with the firewall activated since they need to connect to servers and players.
**Security leak:** When inspecting the packets a security issue was found. Quiz Battle sends readable login information which should definitely be encrypted. The login information can be seen in Figure 5.6. Since the account name is an email there is a strong likelihood that the same password is used in the email account and the Quiz Battle account. This means that if someone gets access to the Quiz Battle account that person could also access the email which would create further violations of security. The risk that this information gets in the wrong hands and that it can be used to cause damage is probably small but these kinds of information leaks still should not occur in today's games.

![Hypertext Transfer Protocol](image)

**Figure 5.6. Account information in plain text for Quiz Battle.**
5.4 Tower Madness vs. Elemental Tower Defense

This section describes a detailed comparison of Tower Madness and Elemental Tower Defense and shows how similar games can have a very different energy consumption. They are both time-sensitive tower defense games and therefore the gameplay and objectives are very similar. However, they have a very different networking behaviour which is noticeable when looking at the communication energy consumption, the amount of data they send and the time they spend in each RRC-state.

On average, Tower Madness consumes 65 J and sends 4576 KB of data in 15 minutes and Elemental Tower Defense consumes 444 J and sends 73 KB which is a considerable difference. It is interesting that Tower Madness sends a lot more data than Elemental Tower Defense but still consumes a lot less energy. The time spent in each RRC state is also very different for the two games as shown in Figure 5.7.

The big difference in these games is the way ads are handled. Tower Madness fetches ads only when a new level is entered. Elemental Tower Defense on the other hand, fetches and shows the advertisements periodically while the player is in the levels and building towers. In these particular games, each level takes quite a long time to complete, approximately 5-10 minutes.

The ad-behaviour and the long level duration causes a big difference in the RRC state pattern of these games. Since Tower Madness only fetches advertisements when a level is entered and the levels take a long time to complete ads will be fetched with a very low frequency. Figure 5.8 and 5.9 shows the data transmissions of these games. In Figure 5.8, levels are started at the 50 second mark and the 370 second mark. The figure clearly shows that data is transmitted only when levels are entered and the UE can therefore remain in PCH while the levels are played. Since Elemental Tower Defense instead fetches ads during levels the UE cannot spend this time in the PCH state like in Tower Madness. Figure 5.9 shows how the data is sent by Elemental Tower Defense. The red uplink packets are HTTP Get requests for fetching ads.

Figure 5.7. The percentage of time spent in each RRC state of Elemental Tower Defense and Tower Madness.
The figure shows that ads are fetched approximately every 20 seconds and that data is constantly transmitted. This explains why it spends most of the time in the FACH and DCH states, and why it consumes much more energy even though the gameplay is very similar to Tower Madness.

The reason that Tower Madness transmits a lot more data can be explained by the fact that it receives video advertisements with a large file size while Elemental Tower Defense receives advertisements in form of small images.

This comparison shows that the way ads are fetched heavily effects the energy consumed by games and it is something to consider. According to this example, developers should avoid fetching advertisements during levels when levels take a long time to complete, if they are interested in optimising the energy consumption of their games.

Figure 5.8. Data sent by Tower Madness.

Figure 5.9 Data sent by Elemental Tower Defense.


5.5 CPU Utilization

In this section the CPU utilization of the 20 selected games is presented. Figure 5.11 shows the average CPU utilization and standard deviation for each game. The CPU utilization of the games can indicate the overall activity of the system while playing. Utilizing the CPU consumes energy and a higher utilization increases the power consumption of the phone.

![Figure 5.11. CPU utilization of the selected games.](image)

Figure 5.11 shows that there is less varying results among the games when it comes to CPU utilization. The CPU utilization of the games is between 14 % and 51 % but 16 of the games have a CPU utilization of 14–26 %. The standard deviation for most of the games is quite low which indicates that each run produces similar CPU utilization.

The four games with the highest CPU utilization are Angry Birds Go (ABGO), Sonic Dash (SD), School Driving 3D (SD3D) and Arcane Legends (AL). What is common for these games is that they all have 3D graphics. This could indicate that 3D games puts a higher load on the CPU than other games.

The game that really stands out among the selected games is Sonic Dash (SD) with a significant CPU utilization of 51 %. The high utilization of this game should mean that it drains the battery much quicker than the games with a lower utilization. That this game would have a higher utilization was expected since it has a very fast-paced gameplay and very advanced graphics.
5.6 System Call Quantification

This section presents the information gathered from the system call traces that were collected with Strace. The section provides an overview of the most common system calls and provides a quantification of their occurrences. This information can help researchers working in the security area because system call behaviour of games can be used by malware detection systems. This section will simply analyse the system calls of Android games, and will not focus on security issues.

A system call is how a program requests a service from an operating system's kernel. This can include hardware related services, creation and execution of new processes and communication with integral kernel services such as process scheduling.

These games can be considered to be benign applications since they are very popular games and were downloaded from the official market.

Figure 5.12 shows the number of system call instances in the 20 games that were selected. A brief description of the most common system calls can be found in Appendix C.

Quite a large amount of system calls appeared in the 20 games. 56 different system calls appeared in at least one of the twenty games and 19 were common for all the games.

Other studies have focused on system calls related to mobile applications to try and determine ways of detecting malicious versions. As mentioned in Section 1.6 in one of the related works, certain system calls only occurred in the malicious applications of their study [9]. From our analysis we can see that some of these specific system calls frequently occur in mobile games. Recv, recvfrom and socket, which are network related system calls were used by all of the 20 games we selected. Fdatasync, mkdir, rename and statfs64 occurred in some of the games.
Figure 5.12. The number of games where a system call was observed.
Table 5.1 shows the total volume of system calls made, the systems calls made per second and the number of different system calls that occurred in each of the 20 games.

<table>
<thead>
<tr>
<th>Game</th>
<th>Total system calls</th>
<th>Syscall intensity (Syscalls/Second)</th>
<th>Number of different system calls</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angry Birds</td>
<td>402238</td>
<td>447</td>
<td>31</td>
</tr>
<tr>
<td>Angry Birds RIO</td>
<td>497444</td>
<td>553</td>
<td>30</td>
</tr>
<tr>
<td>Angry Birds Starwars 1</td>
<td>455774</td>
<td>506</td>
<td>27</td>
</tr>
<tr>
<td>Angry Birds Starwars 2</td>
<td>473815</td>
<td>526</td>
<td>28</td>
</tr>
<tr>
<td>Angry Birds GO</td>
<td>523222</td>
<td>580</td>
<td>31</td>
</tr>
<tr>
<td>Cut the Rope 1</td>
<td>408904</td>
<td>454</td>
<td>46</td>
</tr>
<tr>
<td>Cut the Rope 2</td>
<td>127449</td>
<td>141</td>
<td>33</td>
</tr>
<tr>
<td>Candy Crush Saga</td>
<td>164093</td>
<td>182</td>
<td>20</td>
</tr>
<tr>
<td>Candy Crush Soda Saga</td>
<td>138848</td>
<td>154</td>
<td>30</td>
</tr>
<tr>
<td>Bubble Witch Saga 1</td>
<td>373958</td>
<td>415</td>
<td>20</td>
</tr>
<tr>
<td>Bubble Witch Saga 2</td>
<td>462615</td>
<td>514</td>
<td>37</td>
</tr>
<tr>
<td>Elemental Tower Defense</td>
<td>362150</td>
<td>402</td>
<td>23</td>
</tr>
<tr>
<td>Tower Madness</td>
<td>467872</td>
<td>520</td>
<td>34</td>
</tr>
<tr>
<td>Sonic Dash</td>
<td>96101</td>
<td>107</td>
<td>42</td>
</tr>
<tr>
<td>Race the Traffic</td>
<td>179149</td>
<td>200</td>
<td>31</td>
</tr>
<tr>
<td>School Driving 3D</td>
<td>168260</td>
<td>187</td>
<td>38</td>
</tr>
<tr>
<td>Empire Four Kingdoms</td>
<td>3568263</td>
<td>3965</td>
<td>39</td>
</tr>
<tr>
<td>Quiz Battle</td>
<td>2586039</td>
<td>2873</td>
<td>38</td>
</tr>
<tr>
<td>Star Legends</td>
<td>901722</td>
<td>1002</td>
<td>21</td>
</tr>
<tr>
<td>Arcane Legends</td>
<td>853285</td>
<td>948</td>
<td>19</td>
</tr>
</tbody>
</table>

Table 5.1. Kernel level activity.

The number of different system calls made by the games range between 19 and 46 and on average 31 system calls were made. 9 games used less than the average and 11 games used more than the average.

The system call intensity of the games vary greatly between approximately 100 to 4000 systems calls per second.

The percentage of time spent in each system call is almost identical for the games. About 98 % of the time spent on system calls for all games is spent in epoll_wait, which is a system call that waits for an I/O event on a file descriptor. Most of the calls that are made relate to the system calls that are common, and appear in all of the games. The system calls that are rare and only occur in few of the games are only called a very few times.
Chapter 6

Conclusions

In this project we have profiled 20 free Android games by performing a resource study focusing on the 3G transmissions, CPU usage, and the volume of system calls they perform.

We evaluated the automatic game player Monkey Gamer to determine if it could be used for trace collection in this thesis. We have seen that using Monkey Gamer would have limited the game selection because it cannot play all types of games. We have also shown that the energy consumption of the games are different when Monkey Gamer and humans play. Having the games played by humans instead resulted in a more interesting game selection as we were able to select both online and time-sensitive games, and also ensured that the traces collected are realistic. Improving the object-detection system to improve the speed of Monkey Gamer could allow it to play more types of games and produce more human-like traces.

The results show that the communication energy of the free Android games differs a lot, being generally high. The results also show that there is a big difference in the amount of data that is transmitted by the games but that the amount of data is not as important as how the data is sent.

Our study further shows that the type of game does not correlate with the expected energy consumption. For example, the single player games which are expected to consume less than online games resulted in similar consumption. Due to the nature of online games, we expect them to spend a lot of time in the high energy states, which is what we have seen in our analysis. However, we believe it is hard to justify the amount of time single player games spend in the higher power states. The problem is mainly caused by the fact that the games make use of in-game advertisements and statistics which are transmitted over the network quite frequently. Since data is sent often the UE does not have time to idle. We believe that the situation could be improved and the energy consumption significantly reduced by prefetching several ads with large intervals instead of one at a time with small intervals. The level related statistics could be aggregated and sent together reducing the radio active time. This would mean that the data can be sent in bursts to allow the UE to spend more time idle.

The comparison between Tower Madness and Elemental Tower Defense is a good example of how handling the ads in different ways can create a great difference in energy consumption. This shows that the way advertisements in games are handled (eg., ad-fetching frequency and where ads are fetched in the game) is very important when it comes to energy consumption. Perhaps this finding can make game developers more aware of the situation and perhaps it can influence them to think about how the use of in-game advertisements impacts the battery life time of the
consumers devices. We believe that developers could improve their applications more from the communication energy perspective.

Moreover, the advertisements are not attractive from the player perspective since they tend to interrupt the gameplay and as we mentioned, reduce the battery lifetime. Currently, users can avoid this energy consumption by either disconnecting from the network while playing games on smartphones or activating a firewall that blocks the games network access.

We also analysed the kernel-level activity of the games by showing which the most common system calls are and quantified the occurrences of them. We have seen that the system call intensity of the games vary a lot although they typically have a high intensity. We also saw that the system call behaviour was similar for many of the games because they mostly make use of similar system calls.

EnergyBox was used in this project to estimate the communication energy consumed by the games as well as extracting RRC information such as state time distribution and amount of state transitions. Estimating the communication energy using EnergyBox was very advantageous because the estimated energy is completely isolated to the energy consumed by the network interface since it makes use of packet traces collected while the games are played.

6.1 Future Work

Here we present topics that could be interesting in future projects.

- Extend the system call analysis towards malware detection using potential malware versions of the same games.
- Extend the work by analysing in detail the transmissions and characterise the traffic patterns by investigating the inter-packet intervals and packet sizes of games.
- Optimise the networking transmissions of an open source game to analyse the best prefetching and batching strategies.
- Compare our findings with traces from paid version of the games using 3G, to see how much energy can be saved by avoiding ads.
- Translate the CPU utilization of the games to power consumption.
- Analyse the GPU usage of a smartphone while playing Android Games for a more refined picture of processing energy usage.
Bibliography


http://www.appbrain.com/stats/number-of-android-apps


http://www.linuxmanpages.com/man2/


Appendix A, Classification by user input.

The following table shows the input type of the top 100 free games in the Google Play Store. The table is used to determine the percentage of games that can be played by Monkey Gamer. The table also shows if a certain game can be selected for analysis if Monkey Gamer is used for trace collection.

<table>
<thead>
<tr>
<th>Top</th>
<th>Name</th>
<th>Input Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Don't Step on the White Block</td>
<td>Time Independent</td>
</tr>
<tr>
<td>2</td>
<td>100 Ballz</td>
<td>Time Dependent</td>
</tr>
<tr>
<td>3</td>
<td>Geometry Dash Lite</td>
<td>Time Dependent</td>
</tr>
<tr>
<td>4</td>
<td>Candy Crush Saga</td>
<td>Time Independent</td>
</tr>
<tr>
<td>5</td>
<td>Monster Warlord</td>
<td>Time Independent</td>
</tr>
<tr>
<td>6</td>
<td>Farm Heroes Saga</td>
<td>Time Independent</td>
</tr>
<tr>
<td>7</td>
<td>Don't Tap the White Tile</td>
<td>Time Dependent</td>
</tr>
<tr>
<td>8</td>
<td>Forest Mania</td>
<td>Time Independent</td>
</tr>
<tr>
<td>9</td>
<td>Transparent Screen Free</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>Hay Day</td>
<td>Time Dependent</td>
</tr>
<tr>
<td>11</td>
<td>Quizkampen</td>
<td>Intelligence</td>
</tr>
<tr>
<td>12</td>
<td>Make It Rain: Love of Money</td>
<td>Time Dependent</td>
</tr>
<tr>
<td>13</td>
<td>Simulator Laser</td>
<td>-</td>
</tr>
<tr>
<td>14</td>
<td>Subway Surfers</td>
<td>Time Dependent</td>
</tr>
<tr>
<td>15</td>
<td>Clash of Clans</td>
<td>Time Dependent</td>
</tr>
<tr>
<td>16</td>
<td>Wordfeud FREE</td>
<td>Intelligence</td>
</tr>
<tr>
<td>17</td>
<td>Fifa 14 by EA SPORTS</td>
<td>Time Dependent</td>
</tr>
<tr>
<td>18</td>
<td>2048 Number Puzzle game</td>
<td>Time Independent</td>
</tr>
<tr>
<td>19</td>
<td>Pepi Skate 3D</td>
<td>Time Dependent</td>
</tr>
<tr>
<td>20</td>
<td>FarmVille 2: Livet på landet</td>
<td>Time Dependent</td>
</tr>
<tr>
<td>21</td>
<td>3D Ball FREE</td>
<td>Time Dependent</td>
</tr>
<tr>
<td>22</td>
<td>School Driving 3D</td>
<td>Time Dependent</td>
</tr>
<tr>
<td>23</td>
<td>Trials Frontier</td>
<td>Time Dependent</td>
</tr>
<tr>
<td>24</td>
<td>Let's Create! Pottery Live</td>
<td>Time Dependent</td>
</tr>
<tr>
<td>25</td>
<td>Ultimate Block Puzzle</td>
<td>Time Independent</td>
</tr>
<tr>
<td>26</td>
<td>Elements Epic Heroes</td>
<td>Time Dependent</td>
</tr>
<tr>
<td>27</td>
<td>Paperama</td>
<td>Time Independent</td>
</tr>
<tr>
<td>28</td>
<td>Pet Rescue Saga</td>
<td>Time Independent</td>
</tr>
<tr>
<td></td>
<td>Game Title</td>
<td>Time Dependency</td>
</tr>
<tr>
<td>---</td>
<td>----------------------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>29</td>
<td>Hill Climb Racing</td>
<td>Time Dependent</td>
</tr>
<tr>
<td>30</td>
<td>Brave Frontier RPG</td>
<td>Time Independent</td>
</tr>
<tr>
<td>31</td>
<td>My Talking Tom</td>
<td>Time Independent</td>
</tr>
<tr>
<td>32</td>
<td>Temple Run 2</td>
<td>Time Dependent</td>
</tr>
<tr>
<td>33</td>
<td>Angry Birds</td>
<td>Time Independent</td>
</tr>
<tr>
<td>34</td>
<td>Game of War – Fire Age</td>
<td>Time Dependent</td>
</tr>
<tr>
<td>35</td>
<td>Happy Forest</td>
<td>Time Independent</td>
</tr>
<tr>
<td>36</td>
<td>Impossible Jump</td>
<td>Time Dependent</td>
</tr>
<tr>
<td>37</td>
<td>Bubble Witch Saga 2</td>
<td>Time Independent</td>
</tr>
<tr>
<td>38</td>
<td>Worldcraft 2</td>
<td>Time Dependent</td>
</tr>
<tr>
<td>39</td>
<td>Fruit Ninja Free</td>
<td>Time Dependent</td>
</tr>
<tr>
<td>40</td>
<td>Cut the Rope 2</td>
<td>Time Independent</td>
</tr>
<tr>
<td>41</td>
<td>Pou</td>
<td>Time Independent</td>
</tr>
<tr>
<td>42</td>
<td>Transporter Plane 3D</td>
<td>Time Dependent</td>
</tr>
<tr>
<td>43</td>
<td>LEGO DUPLO Glass</td>
<td>Time Dependent</td>
</tr>
<tr>
<td>44</td>
<td>DUMMA MEJ</td>
<td>Time Dependent</td>
</tr>
<tr>
<td>45</td>
<td>Balance 3D</td>
<td>Time Dependent</td>
</tr>
<tr>
<td>46</td>
<td>Hungry Shark Evolution</td>
<td>Time Dependent</td>
</tr>
<tr>
<td>47</td>
<td>Ruzzle Free</td>
<td>Time Independent</td>
</tr>
<tr>
<td>48</td>
<td>Polis mot Tjuv</td>
<td>Time Dependent</td>
</tr>
<tr>
<td>49</td>
<td>Tiny Dice Dungeon</td>
<td>Time Independent</td>
</tr>
<tr>
<td>50</td>
<td>Rörmokare</td>
<td>Time Independent</td>
</tr>
<tr>
<td>51</td>
<td>Racing Rivals</td>
<td>Time Dependent</td>
</tr>
<tr>
<td>52</td>
<td>Watch_Dogs Companion:ctOS</td>
<td>Time Independent</td>
</tr>
<tr>
<td>53</td>
<td>Stickman Trampoline FREE</td>
<td>Time Dependent</td>
</tr>
<tr>
<td>54</td>
<td>Benji Bananas</td>
<td>Time Dependent</td>
</tr>
<tr>
<td>55</td>
<td>Castle Clash</td>
<td>Time Independent</td>
</tr>
<tr>
<td>56</td>
<td>Clash of Lords 2</td>
<td>Time Dependent</td>
</tr>
<tr>
<td>57</td>
<td>Ambulance Doctor</td>
<td>Time Independent</td>
</tr>
<tr>
<td>58</td>
<td>Megapolis</td>
<td>Time Dependent</td>
</tr>
<tr>
<td>59</td>
<td>Dumb Ways to Die</td>
<td>Time Dependent</td>
</tr>
<tr>
<td>60</td>
<td>Archery tournament</td>
<td>Time Independent</td>
</tr>
<tr>
<td>61</td>
<td>Moy 2</td>
<td>Time Independent</td>
</tr>
<tr>
<td>62</td>
<td>EmojiNation</td>
<td>Intelligence</td>
</tr>
<tr>
<td>63</td>
<td>Flow Free</td>
<td>Time Independent</td>
</tr>
<tr>
<td>64</td>
<td>The Sims</td>
<td>Time Dependent</td>
</tr>
<tr>
<td></td>
<td>Game Title</td>
<td>Category</td>
</tr>
<tr>
<td>---</td>
<td>------------------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>65</td>
<td>Papa Pear Saga</td>
<td>Time Independent</td>
</tr>
<tr>
<td>66</td>
<td>8 Ball Pool</td>
<td>Time Independent</td>
</tr>
<tr>
<td>67</td>
<td>Jetpack Joyride</td>
<td>Time Dependent</td>
</tr>
<tr>
<td>68</td>
<td>Bloons TD Battles</td>
<td>Time Dependent</td>
</tr>
<tr>
<td>69</td>
<td>Sonic Dash</td>
<td>Time Dependent</td>
</tr>
<tr>
<td>70</td>
<td>Lego Juniors Quest</td>
<td>Time Dependent</td>
</tr>
<tr>
<td>71</td>
<td>Angry Birds Go!</td>
<td>Time Dependent</td>
</tr>
<tr>
<td>72</td>
<td>Solitair</td>
<td>Time Independent</td>
</tr>
<tr>
<td>73</td>
<td>Football Quiz</td>
<td>Intelligence</td>
</tr>
<tr>
<td>74</td>
<td>Wake Up! Penguins</td>
<td>Time Independent</td>
</tr>
<tr>
<td>75</td>
<td>Gunship Battle: Helicopter 3D</td>
<td>Time Dependent</td>
</tr>
<tr>
<td>76</td>
<td>Empire: Four Kingdoms</td>
<td>Time Dependent</td>
</tr>
<tr>
<td>77</td>
<td>Little Hair Doctor</td>
<td>Time Dependent</td>
</tr>
<tr>
<td>78</td>
<td>Candy Blast Mania</td>
<td>Time Independent</td>
</tr>
<tr>
<td>79</td>
<td>Plants vs Zombies 2</td>
<td>Time Dependent</td>
</tr>
<tr>
<td>80</td>
<td>Frozen Free Fall</td>
<td>Time Independent</td>
</tr>
<tr>
<td>81</td>
<td>Bike Race Free</td>
<td>Time Dependent</td>
</tr>
<tr>
<td>82</td>
<td>Smash Hit</td>
<td>Time Dependent</td>
</tr>
<tr>
<td>83</td>
<td>Princess Back Spa</td>
<td>Time Independent</td>
</tr>
<tr>
<td>84</td>
<td>PES Manager</td>
<td>Time Dependent</td>
</tr>
<tr>
<td>85</td>
<td>Deer hunter</td>
<td>Time Dependent</td>
</tr>
<tr>
<td>86</td>
<td>Cut the Rope Full Free</td>
<td>Time Independent</td>
</tr>
<tr>
<td>87</td>
<td>Head Soccer</td>
<td>Time Dependent</td>
</tr>
<tr>
<td>88</td>
<td>The Simpsons: Tapped Out</td>
<td>Time Independent</td>
</tr>
<tr>
<td>89</td>
<td>Perfect Piano</td>
<td>Time Dependent</td>
</tr>
<tr>
<td>90</td>
<td>One Touch Drawing</td>
<td>Time Independent</td>
</tr>
<tr>
<td>91</td>
<td>Flight Simulator: Fly Plane 3D</td>
<td>Time Dependent</td>
</tr>
<tr>
<td>92</td>
<td>God of Light</td>
<td>Time Independent</td>
</tr>
<tr>
<td>93</td>
<td>Criminal Legacy</td>
<td>Time Independent</td>
</tr>
<tr>
<td>94</td>
<td>Dragon City</td>
<td>Time Independent</td>
</tr>
<tr>
<td>95</td>
<td>Sonic Racing Transformed</td>
<td>Time Dependent</td>
</tr>
<tr>
<td>96</td>
<td>Plague Inc.</td>
<td>Time Independent</td>
</tr>
<tr>
<td>97</td>
<td>Radio Apan - Banankalas</td>
<td>Time Independent</td>
</tr>
<tr>
<td>98</td>
<td>Fast and Furious 6: The Game</td>
<td>Time Dependent</td>
</tr>
<tr>
<td>99</td>
<td>Striker Soccer 2</td>
<td>Time Dependent</td>
</tr>
<tr>
<td>100</td>
<td>Godzilla: Strike Zone</td>
<td>Time Dependent</td>
</tr>
</tbody>
</table>
Appendix B, Links to Google Play Store

This appendix contains screenshots and links to each game's Google Play Store page.

**Angry Birds**

**Angry Birds RIO**

**Angry Birds Go**

**Angry Birds Starwars**
AngryBirds Starwars 2

Cut The Rope 1 & 2

Bubble Witch Saga

Bubble Witch Saga 2
Candy Crush Saga

Candy Crush Soda Saga

Race The Traffic

School Driving 3D
Elemental Tower Defense

Star Legends
https://play.google.com/store/apps/details?id=sts.sl

Arcane Legends
https://play.google.com/store/apps/details?id=sts.al

QuizBattle
Empire Four Kingdoms

Sonic Dash

Tower Madness
Appendix C, System call descriptions

These descriptions are taken from the unix man pages [18].

**epoll_wait**
epoll_wait waits for an I/O event on an epoll file descriptor.

**futex**
futex is used for fast user-space locking. The call provides a method for a program to wait for a value at a given address to change, and a method to wake up anyone waiting on a particular address.

**ioctl**
ioctl manipulates the underlying device parameters of special files. In particular, many operating characteristics of character special files (e.g., terminals) may be controlled with ioctl requests.

**clock_gettime**
clock_gettime is used to get the current time of a specified clock.

**getpid**
getpid returns the process ID of the calling process.

**recv and recvfrom**
recv and recvfrom calls are used to receive messages from a socket, and may be used to receive data on a socket.

**munmap**
munmap is used to map or unmap files or devices into memory.

**sigprocmask**
sigprocmask is used to fetch and/or change the signal mask of the calling thread. The signal mask is the set of signals whose delivery is currently blocked for the caller.

**dup**
dup is used to duplicate a file descriptor
clone
clone is used to spawn child processes.

sched_yield
A process can relinquish the processor voluntarily without blocking by calling sched_yield. The process will then be moved to the end of the queue for its static priority and a new process gets to run.

fchown32
fchown changes ownership of a file.

fsync and fdatasync
Both fsync and fdatasync are used to synchronize a file's in-core state with storage device.

statfs64
statfs64 is used to get file system statistics.

umask
The umask is used by the system call open to set initial file permissions on a newly-created file.

read
read is used to read from a file descriptor.

write
write is used to write to a file descriptor.

open
open is used to open or possibly creates a file.