Addis Ababa Institute of Technology

School of Graduate Studies

Faculty of Technology

Department of Electrical and Computer Engineering

MOBILE OPERATING SYSTEM FEATURE COMPARISION AND DESINGING BENCHMARK TO EVALUTE THEIR PERFORMANCE

By: Haymanot Minalu Ayele

A Thesis submitted to the School of Graduate Studies of Addis Ababa Institute of Technology, in partial fulfillment of the requirements for the Degree of Master of Science in Computer Engineering.

February, 2011

Addis Ababa, Ethiopia

Addis Ababa Institute of Technology

School of Graduate Studies

Faculty of Technology

Department of Electrical and Computer Engineering

MOBILE OPERATING SYSTEM FEATURE COMPARISION AND DESINGING BENCHMARK TO EVALUTE THEIR PERFORMANCE

By: Haymanot Minalu Ayele

Advisor: Prof. Dr. Sayed Nouh

February 2011

Addis Ababa Institute of Technology

School of Graduate Studies

DECLARATION

I, the undersigned, hereby declare that this thesis is my original work performed under the supervision of Prof. Sayed Nouh, has not been presented as a thesis for a degree program in any other university and all sources of materials used for the thesis are duly acknowledged.

Name: Haymanot Minalu Ayele

Signature: _____

Place: Addis Ababa

Date: February 2011

This thesis has been submitted for examination with my approval as a university advisor.

Advisor's Name: Prof. Dr. Sayed Nouh

Signature: _____

Place: Cairo Egypt

Date: 12/2/2011

School of Graduate Studies
Addis Ababa Institute of Technology

MOBILE OPERATING SYSTEM FEATURE COMPARISION AND DESINGING BENCHMARK TO EVALUTE THEIR PERFORMANCE

By: Haymanot Minalu Ayele

Approved by Board of Examiners

Dr. Getahun Mekuria _____

Chairman, Signature

Department Head of Electrical and Computer Engineering

Prof. Sayed Nouh Sayed Nouh

Advisor Signature

Dr. Kumudha Raimond _____

Internal Examiner Signature

Mr. Yoseph Abate _____

External Examiner Signature

ACKNOWLEDGEMENT

First of all I thank The Almighty God for letting me finish this thesis. The painstaking effort exerted on this thesis would not have been successful without His help.

I am so grateful for my advisor Prof. Sayed Nouh for enlightening me to this wonderful Thesis work. From the day of inception to the last day his enthusiastic advise were great asset.

I am indebted to all the staff members of the Department. All of their comments and advises were valuable and greatly appreciated. Dr. Kuhmuda takes the lion share in this regard as her feedbacks on every seminar were indispensable.

Numerous friends and colleagues are on the line to be duly acknowledged; though I cannot mention all, few in alphabetical order are: Amanuel, Bethelehem, and Yenatfanta. All deserve my gratitude in every aspect of their effort and support. Those encouraging smiles and share of ideas were driving me to success.

Finally, but not least my family have always been my aspirations to keep me going with their endless love and care. Sol and Mekdi, I can never be grateful enough for all the push and care, without them I would have been a dropout sometime back.

ABSTRACT

Mobile devices being available everywhere and support many functionalities they attract business areas and customers. With this interest different new technologies are implemented to satisfy user needs. Different operating system developers are in a strong competition to hold major market share by providing best features. As the technology moves forward quality should be tested. Hence standard benchmark to measure basic feature performance of operating systems is needed. The results found from the benchmark measurement are used to identify weak and strong points of the operating system. This thesis work focuses on designing benchmark that measures performance of mobile operating systems. The most common mobile operating systems are Symbian, Android, Windows Mobile, and Blackberry. My first task is studying some features of each operating system. The basic features selected to study the internal behaviour of operating systems were power management, memory management and multitasking. The next task is studying how to design the benchmark. In this work there are two major points: determining metrics to be measured and studying the programming environment that was used to develop the benchmark. The benchmark measured performance of the operating systems using metrics. The metrics are time of completion of the work, battery percent used to complete the work, and memory used during the work. The benchmark was tested on emulator and on actual mobile devices for each operating system. Using the results of the metrics measured a conclusion is drawn and the reason for the weak point of the performance of the operating system is identified. Based on the benchmark result Android, Symbian, Blackberry and Window Mobile operating systems are listed in order of their performance. Mainly the thesis work is used to identify the strong and weak points of each mobile operating system. So this work can be a basis foundation for choosing best suited operating system for an area of interest and also used for further feature improvement design of operating systems.

Keywords: Benchmark Application, Metrics, Features Performance, Operating systems.

TABLE OF CONTENTS

ACKNOWLEDGEMENT ii

ABSTRACT iii

TABLE OF CONTENTS iv

LIST OF TABLES vii

LIST OF ACRONYMS viii

1. INTRODUCTION 1

1.1 Background 1

1.2 Statement of the problem 3

1.3 Objective 4

1.4 Methodology 4

1.5 Scope of the thesis 4

1.6 Thesis outline 5

1.7 Contribution 6

2. LITERATURE REVIEW 7

2.1 Features of mobile operating systems 7

2.2 Metrics used to measure performance 8

2.3 Mobile operating systems internals 12

2.4 Mobile Computing Device Power Management 13

3. MOBILE OPERATING SYSTEMS FEATURES 17

3.1 Android OS 17

3.2 Symbian OS 27

3.3 Blackberry OS 34

3.4 Windows OS 36

4. DESIGN AND IMPLEMENTATION OF BENCHMARK 41
4.1 Software Environments 41
4.2 Design of Benchmark Applications 46
4.3 Metrics measured in the Benchmark 56
4.4 Implementation of the Benchmark 56
5. RESULTS AND DISCUSSION 59
5.1 Android Results 60
5.2 Symbian Results 61
5.3 Windows mobile Results 62
5.4 Blackberry Results 64
6. PERFORMANCE ANALYSIS 65
6.1 Emulator and real device Results 65
6.2 Memory taken by the application 68
6.3 Time taken by the application 71
6.4 Battery percent decreased in running application 72
6.5 Multitasking Ability 73
6.6 Previous work on Benchmark 74
7. CONCLUSION AND RECOMMENDATION 75
7.1 Conclusion 75
7.2 Recommendation 76
REFERENCES 77

LIST OF FIGURES

Figure 1.1: Operating systems Market share in 2010 1

Figure 3.1: High-level look at the Android system architecture 18

Figure 3.2: YAFFS embedded structure 24

Figure 3.3: Android Power Management 26

Figure 3.4: Power Management State Machine (Screen Brightness) 26

Figure 3.5: F32 system architecture 30

Figure 3.6: Typical CPU state transition diagram 32

Figure 3.7: Symbian OS Architecture 33

Figure 3.8 Previous Memory Scheme 39

Figure 3.9: Persistent Memory Storage: WM5 40

Figure 4.1: Android Benchmark Flow chart 49

Figure 4.2: Symbian Benchmark Flow chart 52

Figure 4.3: Blackberry Benchmark Flow chart 54

Figure 4.4: Windows Benchmark Flow chart 55

Figure 5.1: Android emulator results 60

Figure 5.2: Symbian emulator results 61

Figure 5.3: windows Mobile emulator results 63

Figure 5.4: Blackberry simulator results 64

Figure 6.1: Plot of Free RAM versus Time 70

Figure 6.2: Plot of Battery Percent versus Time 73

LIST OF TABLES

Table 2.1: Result of operating system comparison.....8

Table 2.2: Power consumption of various components of the Sharp PC 6785 14

Table 3.1: Wake Lock Settings.....25

Table 6.1: Device specifications.....67

Table 6.2: Memory taken on emulators.....68

Table 6.3: Memory taken on real devices.....69

Table 6.4: Time taken by the application on emulators and real devices.....71

Table 6.5: Battery percent on real devices.....72

LIST OF ACRONYMS

ACPI	Advanced Configuration and Power Management
ADB	Android Debug Bridge
ADT	Android Development Tools
ALP	Access Linux Platform
API	Application Programming Interface
APK	Android Package
APM	Advanced Power Management
ARM	Acorn RISC Machine
BIOS	Basic Input Output System
BSD	Berkeley Software Distribution
DOS	Disk Operating System
EKA2	EPOC Kernel Architecture 2
GUI	Graphical User Interface
IDE	Integrated Development Environment
J2ME	Java 2 Micro Edition
JDE	Java Development Environment
JDK	Java Development Kit
JRE	Java Run Time
MID	Mobile Internal Device
MIDP	Mobile Information Device Profile
MMU	Memory Management Unit
OS	Operating System
PDA	Personal Digital Assistant
PM	Power Management
RAM	Random Access Memory
RIM	Research In Motion
RISC	Reduced Instruction Set Computer
ROM	Read Only Memory
SDK	Software Development Kit
TCP/IP	Transmission Control Protocol /Internet Protocol
UI	User Interface
URL	Uniform Resource Locator
VM	Virtual Machine
WAP	Wireless Application Protocol
WCE	Workload Completion Energy Efficiency
WCM	Workload Completion Memory Efficiency
WCR	Workload Completion Rate
WM6	Windows mobile 6
YAFFS	Yet Another Flash File System

1. INTRODUCTION

1.1 Background

A Mobile operating system, also known as a Mobile OS, a Mobile platform, or a handheld operating system is the operating system that controls a mobile device—similar in principle to an operating system such as Linux or Windows that controls a desktop computer or laptop. However, they are currently somewhat simpler, and deal more with the wireless versions of broadband and local connectivity, mobile multimedia formats, and different input methods.

Like a computer operating system, a mobile operating system is the software platform on top of which other programs run. When you purchase a mobile device, the manufacturer will have chosen the operating system for that specific device. The operating system is responsible for determining the functions and features available on your device, such as thumbwheel, keyboards, WAP, synchronization with applications, e-mail, text messaging and more. The mobile operating system will also determine which third-party applications can be used on your device.

Operating systems that can be found on smart phones include Symbian OS, iPhone OS, RIM's Blackberry, Windows Mobile , Linux, Palm WebOS, Android and Maemo. Common smart phone operating systems are defined below.

Android, WebOS and Maemo are in turn built on top of Linux, and the iPhone OS is derived from the BSD and NeXTSTEP operating systems, which all are related to Unix. The most common operating systems (OS) used in smart phones by 2010 and their market share are Shown in Figure 1.1

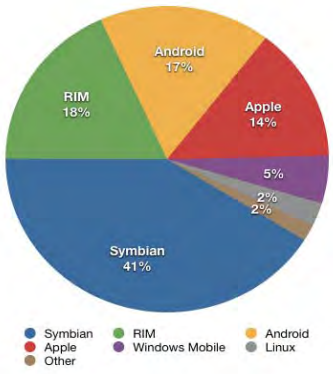


Figure 1.1: Operating systems Market share in 2010

i. **Symbian OS from Symbian Ltd.:** Symbian has the largest share in most markets worldwide, but lags behind other companies in the relatively small but highly visible North American market. This matches the success of its largest shareholder and customer, Nokia. It is used by many major handset manufacturers, including BenQ, LG, Motorola, Samsung, and Sony Ericsson.

ii. **RIM Blackberry OS:** This OS is focused on easy operation and was originally designed for business. Recently it has seen a surge in third-party applications and has been improved to offer full multimedia support.

iii. **IPhone OS from Apple Inc.:** The iPhone uses an operating system called iPhone OS, which is derived from Mac OS. Third party applications were not officially supported until the release of iPhone OS 2.0 on July 11th 2008. Before this, "jail breaking" allowed third party applications to be installed, and this method is still available. But this problem is solved on iPhone OS 3.0, iPhone OS 4.1 which are released in respectively.

iv. **Windows Mobile OS from Microsoft:** The Windows CE operating system and Windows Mobile middleware are widely spread in Asia. The two improved variants of this operating system, Windows Mobile 6 Professional & Windows Mobile 6 Standard, were unveiled in February 2007. In addition Windows Mobile 6.5 and Windows phone 7 are recent versions released in 2010. Windows Mobile benefits from the low barrier to entry for third-party developers to write new applications for the platform. It has been criticized for having a user interface which is not optimized for touch input by fingers; instead, it is more usable with a stylus.

v. **Android OS from Google Inc.:** Android was developed by Google Inc. Its share of the smart phone market is still small because of its recent release date. Android is an Open Source, Linux-derived platform backed by Google, along with major hardware and software developers (such as Intel, HTC, ARM, and eBay, to name a few), that form the Open Handset Alliance. This OS, though very new, already has a cult following among programmers eager to develop apps for its flexible, Open Source, back end. Android promises to give developers access to every aspect of the phone's operation. This lends many to foresee the promise of further growth for the Android platform.

vi. **Palm webOS from Palm Inc. and Palm OS:** Palm webOS is Palm's next generation operating system. Palm Source traditionally used its own platform developed by Palm Inc.

Access Linux Platform (ALP) is an improvement that was launched in the first half of 2007. [2]

Comparing Mobile Phone Operating Systems

Using different features operating systems performance was compared. The most common operating systems which were considered in this work are symbian, Google Android, Blackberry and windows Mobile. Some categories for comparing features of operating systems are Basics, User Interface, Core Functionality and Third-Party Development. [3]

In addition to the above features the operating systems can be identified by the following features Process Management, Memory Management, File Management, I/O Management, Networking, and Protection System. [4]

Benchmarks are fundamentally tools for evaluating and comparing alternatives. The resulting scores, rankings, and output data from the benchmarks must somehow be representative of some real-world system behaviour.

Mobile operating system benchmark should allow comparison of various system-level mobile operating system features. During testing performance of the operating system the effect of the hardware that is in use should be considered.

1.2 Statement of the problem

The Personal Computer and the Internet have found revolutionary ways to connect people, to entertain them and let them exchange information. But none of these is able to reach each person anywhere and anytime like the cell phone does. Based on the company “The Mobile World” in 2007 the global mobile phone usage had exceeded 3.25 billion at the end of 2007 which is equivalent to around half of the worlds population. This shows what a size is behind the brand “cell phone”. Ten years ago nobody would think about a development like this. [5]

With this increasing number of user the technology grows rapidly. But the growth in different features of operating systems should be measured to test their way of growth. Adding new feature does not guaranty performance increase in the operating system. Generally the main problem is that performance of each mobile operating system is not well known and it is not measured based on specific standard. This thesis work can provide clear information about the features of each operating system in comparison with other and using benchmark their performance is evaluated.

1.3 Objective

General Objective

The main objective of this thesis work is Feature comparison and Performance Evaluation of each mobile operating system. Performance evaluation is based on benchmark results found from real mobile device and emulator software.

Specific Objective

i. Studying details of different operating systems with regard to specified features.
ii. Using these features make a comparison between operating systems
iii. Based on appropriate metric, design benchmark suitable for each operating system.
iv. Using the benchmark evaluate the various operating systems by using mobile device with different operating system and software emulators.
v. Analyze the theoretical and practical performance of these systems

1.4 Methodology

Literature survey in the area: There was adequate literature review performed on previous works. For example previous works on, mobile operating system features, mobile operating system internal structure, and platform metrics of mobile operating system was revised.

Designing Benchmark and Implementation: a benchmark suitable for each operating system was designed and implemented on real mobile device and run on emulator software. Accordingly, the best performing and less performing system was identified based on the metrics of the benchmark.

Mechanism of Driving Conclusion: After adequate literature review, each operating system was studied and a benchmark with appropriate metrics was designed and implemented to measure the performance of the operating system. The result found was used to analyze and compare the performance of different operating systems practically.

1.5 Scope of the thesis

The main focus was dealing with performance of different operating systems using Benchmark. The operating systems selected are Android, Symbian, Blackberry and windows. Iphone and Palm operating systems were in the proposal of the thesis. But Benchmark application was not designed for them.

To develop iphone application the SDK is downloaded on apple computer with Mac operating system. After developing application anyone can not load application to iphone mobiles rather it is installed from official site, iTunes store. This is a barrier for developing Iphone Benchmark.

Palm smart phone market is very low compared to others, it is below 2%. In addition the programming language is HTML5.0, which is different from high level languages. With these reasons palm application is not developed.

The benchmark measured performance of the OS using some specified metrics. The metrics are Memory taken, battery percent decreased, time taken and ability of multitasking. The benchmark measured metrics by giving a task to be done by the OS. The task selected was playing audio file in parallel with downloading image file. The benchmark was implemented on devices with the required OS. Using the results found from the benchmark the comparison of the operating systems was done.

1.6 Thesis outline

The remainder of this Thesis report is organized into the following six chapters: Chapter 2 of the report presents a brief overview on literature reviews. This chapter reviews different paper works which are somehow related with the thesis work. Chapter 3 then discusses different features of operating systems. It points out the detail internal structure and property of each operating system. This section also reviews the good and vulnerable areas of the operating systems. Under chapter 4, there is discussion in the Design and implementation of Benchmark application. This section explains different software environments used to develop Benchmark for different operating systems, it also discuss detail design steps of the bench mark and finally the implementation of the benchmark on real device and on software emulators. The results found after implementation of the benchmark are explained and summarized under chapter 5 of this report. Chapter 6 analyzes the tests performed on the proposed operating systems and performance evaluation is done with comparison in detail.
Finally, conclusions and recommendations for future works are presented in chapter 7.

1.7 Contribution

This work has great contribution for different areas. The Benchmark designed can measure basic operating system features that can help to evaluate performance. Memory taken by the application, battery percent decreased because of the work done and time taken by the application are measured. In addition multitasking ability is tested by giving two works simultaneously. Most of the time mobile customers buy devices not based on their operating systems. But operating systems are fundamental part of the device that makes the device to work everything. Testing performance of OS can help customers, manufactures, and developers. Hence for customers who need to get their choice of Mobile operating system with the required feature can first know the best and bad feature of the operating system.

For manufactures of mobile devices they can also compare and choose their best desire of operating system that is going to be implemented in the device

And finally for operating system feature and application developers once the vulnerable point of the system is known they can make an update to satisfy user requirement and to be well-suited.

2. LITERATURE REVIEW

In this chapter, different paper works that are related to the thesis was reviewed in detail. There are three sections in the chapter. Each section discussed work that deals the same idea as the title. In each section there were different papers read. But here one sample paper is chosen for each subtitle.

2.1 Features of mobile operating systems

A thesis work entitled “The Android mobile platform” by Benjamin Speckmann. The thesis is a review paper that gives an introduction to the new mobile platform Android as well as a comparative evaluation with regard to other mobile operating systems.

The key topic of this thesis is the categorization of Android. Therefore it first gives a historical introduction to cell phones and mobile operating systems. Then it describes the main features of Android for a better understanding of this platform. In the theoretical part Android is compared to the mobile operating systems Symbian OS and Windows Mobile. Features and criteria defined in this part is considered and included in the comparison of these systems.
The practical part contains a comparison of the Software Development Kits (SDK) from Android and Symbian OS. In this context a simple application implementation on both systems is realized to support this comparison. Finally an outlook and a conclusion complete the elaboration.

The Android platform, Symbian OS and Windows Mobile were compared with regard to the main criteria for a mobile operating system. Every criterion was explained in detail and applied to the three operating systems. At the end of every criterion classification points were given from one to zero. The best operating system relating to the criterion got one point, the second got half a point and the third zero points. The total sum of points added for each operating system will show which of them is the “best” with regard to the main criteria. The following table shows the results in detail:

Table 2.1: Result of operating system comparison

	Android	Symbian OS	Windows Mobile
Portability	1	0.5	0
Reliability	1	1	1
Connectivity	1	1	1
Product diversity	1	1	1
Open system	1	0.5	0.5
Kernel size	0.5	1	0
Standards	1	0.5	0.5
Security	1	1	1
Special features	1	0	0.5
Result	**8.5**	**6.5**	**5.5**

With regard to the main criteria the new Android mobile platform gets most points. It is the only truly "open system" which makes the major difference. Also the special feature like the Web browser and the virtual machine Dalvik play a major role in this comparison. Standards in connectivity, portability and security are more or less achieved by every operating system in the same way. [5]

2.2 Metrics used to measure performance

A paper entitled with "Mobile Platform Benchmarks; A Methodology for Evaluating Mobile Computing Devices" By: Daniel McKenna" reviews the following concept.

Today's truly mobile computing devices operate in an untethered environment characterized by wireless Internet connectivity and battery-powered operation. This mobile computing paradigm is significantly different from the conventional desktop computing model, and introduces a new set of evaluation criteria for platform characterization and comparison.

2 Literature Review

Truly mobile platform user's requirements and expectations include mobile application compatibility, performance sufficiency, and long battery life, in addition to the traditional peak performance requirement of desktop computing.

Other distinguishing features of truly mobile computing are the application software popular in these environments, the variety of ways mobile users choose to configure and operate the devices, and the tradeoffs that users face in attempting to get useful work done while striving to maximize the operating time on batteries. Truly mobile computing also covers a broad range of platform types, including hand-held Windows PCs, thin-clients, Internet appliances, and PDAs. This large variety of platform types, operating system and application environments, and usage profiles complicates the task of platform benchmarking and comparison.

Computer industry benchmarks for system characterization evolved out of the desktop-workstation server-mainframe environment that demands continuous peak performance. These traditional benchmarks are inadequate to address the requirements of the new mobile computing paradigm. Truly mobile platforms should be gauged with the attributes important to mobile users. The benchmarks appropriate for truly mobile computer products are different than benchmarks appropriate for desktop and desktop-replacement portable systems.

Mobile computer benchmarks should measure performance in combination with the energy consumption and battery life penalty for that performance, and should address the issue of performance sufficiency as well as peak performance. Mobile platform benchmarks should measure performance and energy consumption using real mobile application workloads in mobile systems configured and operating the way users configure and operate them, under battery power.

This paper describes a methodology for developing benchmarks for mobile platforms that address the shortcomings of traditional computer benchmarks and incorporate the criteria above.

Characterizing mobile platform performance requires a different viewpoint than the traditional desktop workstation- server-mainframe model that demands the delivery of continuous peak performance as the single goal of the system.

2 Literature Review

Performance sufficiency and energy efficiency have become the predominant platform requirements for today's battery-powered mobile computing devices. Improvements in mobile system usability must take into consideration the battery life penalty that is the natural consequence of the peak performance model of processing. Mobile systems must deliver performance sufficient to satisfy the application workload and no more. Each increment of performance beyond the instantaneous requirement of the application workload results in an unacceptable degradation in battery life.

The rapidly evolving market for highly mobile internet computing devices is driven by the need to work and communicate in a free-roaming, battery-powered operating mode. The communications and computing technology infrastructure, combined with the rapid growth of an Internet e-commerce business model, is enabling a revolution in mobile computing. Mobile Internet computing platforms are appearing at a rapid pace in a proliferation of new categories, including thin-and-light PCs, mini- and hand-held PCs, mobile thin clients, Internet appliances, and PDAs. Each of these categories addresses a different constellation of user needs, with a variety of features and operating characteristics. Evaluating and comparing these new mobile devices has become a challenge because of the multitude of platform specific operating systems, applications, features, and value attributes. In addition to these complexities, some of the new technologies developed for truly mobile platforms cannot be evaluated accurately by existing industry benchmark methodologies.

Benchmarks, in general, are supposed to allow comparison of alternatives and yield results that correlate with the real-world experience of users. Mobile platform benchmarks should allow comparison of various component and system-level mobile platform solutions, and the results should correlate with real-world and real-user mobile platform operating experience. The core of the mobile platform benchmark methodology is to run real-world mobile x86-compatible application software (workloads) in usage profiles that correspond with the way users typically interact with the software and the mobile platform. A set of benchmark metrics was selected that incorporate energy usage and efficiency, peak performance (throughput), and performance sufficiency in a way that is plausible, measurable, reproducible, and correlates with real-world experience.

The first mobile platform benchmark metric is called Workload Completion Rate (WCR). WCR is a classic application-level peak performance metric. Another common name for this type of metric is throughput.

Workload completion rate is defined as in equation 2.1:

WCR = (Mobile Workload Completed) / (Time to Complete Workload) (2.1)

The measurement unit for WCR is workload units per hour.

The second mobile platform benchmark metric is called Workload Completion Efficiency (WCE). WCE describes an energy consumed-work completed relationship, and may be thought of as the energy efficiency for a given workload. Workload completion efficiency is defined as in equation 2.2:

WCE = (Mobile Workload Completed) / (Energy Consumed to Complete Workload) (2.2)

The measurement unit for WCE is workload units per Watt-hour. The metric is very important for mobile platforms because it measures energy efficiency and directly relates to battery life. WCE provide a much better measure of mobile system operation than WCR (peak performance) alone, because mobile users experience the energy penalty of accelerated system activity as shortened operating life under battery operation.

The mobile platform benchmark metrics were selected to address the relationships between three independent variables: workload, energy consumption, and time. Workload demand characteristics vary widely across the range of mobile applications. The three variables were combined in the metrics to have plausible intuitive meaning, as described above. The metrics have the property that larger values represent relative improvement in the attributes addressed by the metric. For example, a larger WCR (peak performance) value indicates a “faster” device or system. Likewise, a larger WCE value indicates a more energy efficient solution.

In conclusion the methodology and metrics for the mobile platform benchmarks described here are a significant improvement over existing industry standard PC benchmark efforts, particularly with respect to objective mobile platform characterization.

The metrics chosen are extremely plausible, easily measured, and show strong correlation to real-world mobile platform operating characteristics. The metrics can be applied at the component, subsystem, and system level, and are thus extremely useful tools for system comparison, design trade-off analysis, competitive analysis, and system hardware and software performance and energy efficiency tuning. [1]

2.3 Mobile operating systems internals

A book entitled Symbian OS Internals, Real-time Kernel Programming was written by Jane Sales with Andrew Rogers, Andrew Thoelke, Carlos Freitas, and others. This book reviews the following concept.

The latest versions of Symbian OS are based upon Symbian's new real time kernel. This kernel is designed to make phone-related development easier: base-porting will be easier, device driver development will be easier, and software development will be easier.

Symbian OS Internals is a detailed exposition on the new real-time kernel, providing the insights of the software engineers who designed and wrote it. In the main it is an explanatory text which seeks to describe the functioning of the kernel, and, where relevant, to indicate key differences from its predecessor. This book is invaluable for:

-Those who are involved in porting Symbian OS: This book is not a base-porting manual, since this is already provided by Symbian; but it benefits the base-porting engineer by giving him or her more solid understanding of the OS being ported.

-Those writing device drivers: This book provides an in-depth explanation of how Symbian OS drivers work. Device drivers have changed considerably with the introduction of a single code, so this helps fill the knowledge gap for those converting them to the new kernel.

-Those who wish to know how the Symbian OS kernel works: This book provides a detailed commentary on the internals of Symbian OS, providing information for the varied needs of readers, helping: students studying real-time operating systems, middleware programmers to understand the behaviour of underlying systems, systems engineers to understand how Symbian OS compares to other similar operating systems, and, for those designing for high performance, how to achieve it. [6]

2.4 Mobile Computing Device Power Management

The first paper that discusses about the power management is a paper entitled “A Survey of Power Management Techniques in Mobile Computing Operating Systems” by Gregory F. Welch. This paper reviews the following concept. Many factors have contributed to the birth and continued growth of mobile computing, including recent advances in hardware and communications technology. With this new paradigm however come new challenges in computer operating systems development. These challenges include heretofore relatively unusual items such as frequent network disconnections, communications bandwidth limitations, resource restrictions, and power limitations. It is the last of these challenges that is explored in this paper—that is the question of what techniques can be employed in mobile computer operating systems that can reduce the power consumption of today’s mobile computing devices.

Batteries are typically the largest single source of weight in mobile computing devices. While reduction of the physical dimensions of batteries is certainly a possibility, such efforts alone will reduce the amount of charge retained by the batteries. This will in turn reduce the amount of time a user can use the computing device before being forced to re-charge the batteries. Such restrictions tend to undermine the notion of mobile computing. However if one can succeed in reducing the basic consumption of individual components of mobile computing devices, we have the luxury of either reducing the battery dimensions while retaining the original charge characteristics, or increasing the battery charge time while retaining the original battery dimensions, or some combination of both.

Many physical components are responsible for ongoing power consumption in a mobile computing device. For example, Table 2.2 below lists the top power-consuming components of the Sharp PC 6785. This data is indicative of most of today’s mobile computing devices, and highlights the areas requiring attention for power reduction.

Table 2.2: Power consumption of various components of the Sharp PC 6785

Component	Power (Watts)
base system (2MB, 25 MHz CPU)	3.650
hard drive motor	1.100
math co-processor	0.650
floppy drive	0.500
external keyboard	0.490
LCD screen	0.315

Scheduling for Reduced CPU Energy

As was seen previously, the power consumed by the CPU in a mobile computing device is significant. In fact, if the CPU clock frequency and supply voltage can be controlled, linear and quadratic savings in power can be realized.

Disk Drive Power Management

A common scheme for reducing the power consumed by hard drives in mobile computers is to “spin-down” or turn off drives during extended periods of non-use.

Energy Efficient Indexing on Air

The periodic broadcast of data over wireless communication channels can be considered in a sense a supplement to a mobile user’s secondary storage. While accessing local hard drives consumes significant power, the reception of wireless broadcast data is a relatively low-power operation. When appropriate, periodic data broadcasts can be used to disseminate large volumes of data, e.g. a large common database, to an easily scalable population of listeners.

The advent of mobile computing has introduced new challenges for the designers of computers and computer operating systems. One such significant challenge is to reduce power consumption in ways that adversely affect the user the least.

Savings in power consumption can be “spent” by extending battery life, and (or) reducing the physical size of batteries thus reducing the physical size of mobile computing devices.
In this paper several proposed power conservation techniques are briefly surveyed. These techniques together focus on what are typically the main culprits in terms of power consumption in mobile computing: CPU/memory devices, secondary storage devices such as hard drives, and wireless communication devices used to supplement secondary storage where appropriate. While hardware advances have and will likely continues to reduce the power consumption of these devices, it has seen that efficient operating system techniques can significantly reduce power consumption without considerably affecting the perceived performance. [7]

A second paper by Marc A. Viredaz and others entitled with “Energy Management on Handheld Devices” is also revised in this thesis work.
Handheld devices are becoming ubiquitous and, as their processing and memory capacity increases, they are starting to displace laptop computers for certain tasks -much as laptop computers have displaced desktop computers in many roles. Current handheld devices are evolving either from the organizer or PDA side or from the cellular phone side and both lines are merging rapidly.
However, whatever their origin, all handheld devices share the same Achilles' heel, the battery. This is because battery technology is not improving at the same pace as the energy requirements of handheld electronics. Therefore, energy management, once in the realm of desired features, has become an important design requirement and one of the greatest challenges in portable computing, and it will remain so for a long time to come.

The disparity between energy requirements and energy sources in handheld devices is the result of several factors. First, users expect more functionality from their devices (more performance and memory, better displays, wireless capabilities, etc.), and although partially offset by improvements in low-power electronics, this increased functionality carries corresponding increases in energy consumption. Second, as a consequence of their increased functionality, these handheld devices are displacing other pieces of equipment and as a result are seeing more use between battery charges. Finally, battery technology has not improved at the same pace as the increase in energy requirements.

2 Literature Review

In this paper Energy usage in current handheld devices is studied. In order to do useful energy management, it is important to understand how and where the energy is used in handheld devices.

Energy-saving techniques are also discussed in this paper. Different Energy-saving techniques for Processor, Memory, Display, Audio system, and Wireless networking are studied.

As people are increasingly relying on handheld devices, good energy management is becoming necessary to squeeze the most out of a battery. A plethora of ideas have been proposed and tested by researchers in academia and in industry, but few have made their way into commercial products.

The field is still in its infancy and interesting ideas continue to blossom. Power/energy studies, which have been sadly lacking, are necessary to understand how energy is actually used. Very little data is available and more published studies are badly needed. Another requirement is better tools to track how energy is used, in particular the capability to evaluate power consumption in real time. There is also a chasm in that power measurements have traditionally been in the hands of hardware engineers, while energy management is usually the responsibility of software developers. Hardware engineers need to provide energy-tracking tools that software developers can easily use. At the same time, software engineers need to acquire a better understanding of how the hardware uses energy. In many respects, the energy-management field is at the same stage as performance evaluation decades ago, when a similar gulf existed between hardware engineers who understood ”where cycles were being used” and software engineers who wrote compilers and applications. Perhaps within a few years ”energy-management engineers” will be as numerous as performance engineers are today and will help bridge this still yawning gap.[8]

3. MOBILE OPERATING SYSTEMS FEATURES

3.1 Android OS

Android OS is easy to think of Android as being yet another operating system for high-end mobile phones. It is really a software platform, rather than just an OS, that has the potential to be utilized in a much wider range of devices. In practical terms, Android is an application framework on top of Linux, which facilitates its rapid deployment in many domains.

A key to its likely success is licensing. Android is open source and a majority of the source is licensed under Apache2, allowing adopters to add additional proprietary value in the Android source without source distribution requirements.

When developers want to deploy Linux for embedded applications, Android is the enabler for a broad application developer base, a complete stack on top of the Linux kernel.

Android History: Although Android is quite new technology, it does have a history. It really began in 2005 when Google acquired Android Inc., which started rumours that Google had interests in mobile telephony. The Android product was announced, along with the formation of the Open Handset Alliance in 2007. The following year saw the first Android phone launched and the declaration of Android code as being open source.

Even though Android was created for handsets, many developers began to see a great opportunity to develop other kinds of innovative devices on the Android platform. Significant optimizations and additions would be required, however, to optimize Android for other connected devices.

Android Architecture: An Android system is a stack of software components. At the bottom of the stack is Linux – Linux 2.6 with approximately 115 patches. This provides basic system functionality like process and memory management and security. Also, the kernel handles all the things that Linux is really good at such as networking and a vast array of device drivers, which take the pain out of interfacing to peripheral hardware.

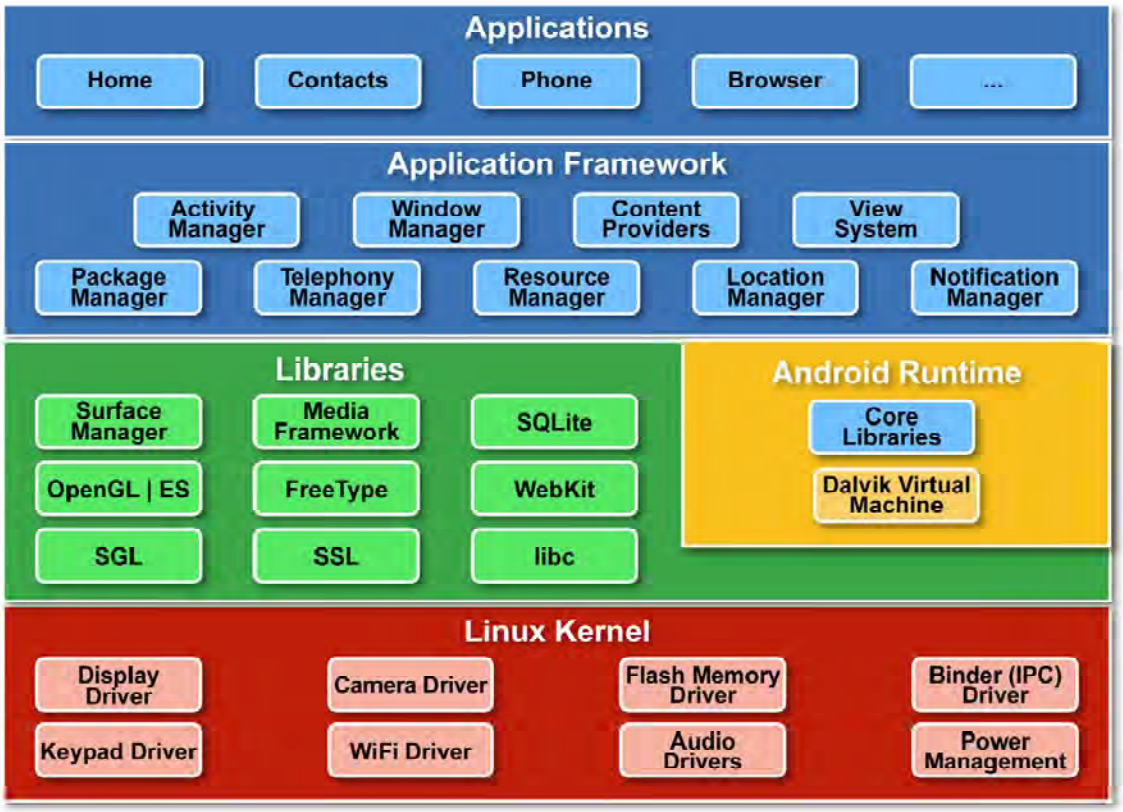


Figure 3.1: High-level look at the Android system architecture.

On top of Linux is a set of libraries including bionic (the Google libc), media support for audio and video, graphics and a lightweight database, which is a useful repository for storage and sharing of application data.

A key component of an Android system is the runtime – the Dalvik VM. This is not strictly a Java virtual machine. It was designed specifically for Android and is optimized in two key ways. It is designed to be instantiated multiple times – each application has its own private copy running in a Linux process. It is also designed to be very memory efficient, being register based (instead of being stack based like most Java VMs) and using its own byte code implementation. The Dalvik VM makes full use of Linux for memory management and multi-threading, which is intrinsic in the Java language.

It is important to appreciate that Android is not a Java virtual machine, but does use the Java language.

The Application Framework provides many higher-level services to applications in the form of Java classes. This will vary in its facilities from one implementation to another.
A key Android capability is the sharing of functionality. Every application can export functionality for use by other applications in the system, thus promoting straightforward software re-use and a consistent user experience.

At the top of the Android software stack are applications. As mentioned, each application may also expose some of its functionality for use by others. For example, the message sending capability of the SMS application can be used by another application to send text messages.
Although there are other options, Android applications are commonly implemented in Java utilizing the Dalvik VM. Not only is the Dalvik highly efficient, but it also accommodates interoperability which results in application portability.

Programming Model: An Android application consists of a number of resources which are bundled into an archive – an Android package. Programs are generally written in Java, built using the standard Java tools, and then the output file is processed to generate specific code for the Dalvik VM. Each application runs in its own Linux process – an instantiation of the Dalvik VM – which protects its code and data from other applications. Of course, there are mechanisms for applications to transfer, exchange, and share data.

An application is a set of components which are instantiated and run as required. There is not really an entry point or main () function.

There are four types of application component: activities, services, broadcast receivers, and content providers.
An Activity is a functional unit of the application, which may be invoked by another activity. It has a user interface of some form. An application may incorporate a number of activities. One activity may be nominated as the default which means that it may be directly executed by the user.

A Service is similar to an activity, except that it runs in the background without a UI. An example of a service might be a media player that plays music while the user performs other tasks.

Broadcast Receivers simply respond to broadcast messages from other applications or from the system. For example, it may be useful for the application to know when a picture has been taken. This is the kind of event that may result in a broadcast message.

A Content Provider supplies data from one application to others on request. Such requests are handled by the methods of the ContentResolver class. The data may be stored in the file system, the database or somewhere else entirely.

When developing an Android application, it is needed to describe it to the system and this is achieved by means of a manifest file. This is an XML file called AndroidManifest.xml which is stored in the root folder of the application's file system.

When an Android application wishes to obtain some functionality from another application or from the system, it can issue an Intent. This is an asynchronous message that is used to activate an activity, service, or broadcast receiver. For an activity or service, the specific action and location of data is included.

Although an intent may include the specific activity required, it can be more generalized and the request resolved by the system. This mechanism is governed by Intent Filters. These filters specify what kind of intents the activities and services of an application can handle. They are described in the manifest file. [9]

Target Architecture: The Linux kernel supports many different target architectures. However only two are fully supported by Android at this time: x86 and ARM. The x86 architecture for Android is mainly targeted at Mobile Internet Devices (MIDs) whereas the ARM platform is prevalent on mobile phones. These architectures are typically used for very different computer systems. The x86 platform is used for general purpose desktop /laptop /server computing whereas ARM is widely in use on mobile devices.

The ARM processor's existing market share of the mobile phone market combined with an emphasis on small code size and low-power operation is the main reason for its widespread usage on Android phones. This is a strong contrast to most desktop/laptop/server Linux systems, which commonly use x86 processors instead.

Kernel Modifications: Android is based on the Linux, but does not use a standard Linux kernel. The kernel enhancements of Android include alarm driver, ashmem (Android shared memory driver), binder driver (Inter-Process Communication Interface), power management, low memory killer, kernel debugger and logger. All these kernel enhancements have been contributed back to the open source community under the GNU Public License (GPL). Alarm driver: provides timers to wake devices up from sleep, ashmem: allows applications to share memory and manages the sharing in kernel levels, binder driver: facilitates inter-process communication since data can be shared by multiple applications through the use of shared memory. A service registered as an IPC service does not have to worry about different threads because binder will handle, monitor and manage them. Binder also takes care of synchronization between processes, and power management: built on the top on standard Linux Power Management (PM) and take a more aggressive policy to manage and save power.

Bionic Standard C Library: On most Linux distributions the GNU C library is used to provide the library routines specified by the ISO C standard for C language programs. Many developers view the GNU C library as being inappropriate for memory constrained platforms such as embedded systems. Furthermore, this library is licensed under the GNU Lesser Public License and therefore restricts licensing of derivative works. These concerns led the developers of Android to instead create their own C library called "Bionic". This library was designed to have fast execution paths, avoid edge cases and remain a simple implementation.
It is composed partly from the BSD C library combined with Android original source code. This results in a combination of the BSD and Android licenses covering the entire library.

The divergent goals of the Android development team drove them to create a custom implementation of the Standard C library. This library is especially suited to operate with the limited CPU and memory available on Android platforms. Furthermore, special security provisions were made in order to ensure the integrity of the system.

Dalvik Virtual Machine: Many of the top cell phone manufacturers such as Nokia, Motorola and Samsung include a mobile optimized version of the Java virtual machine called Java 2, Micro Edition (J2ME). In contrast to these vendors Android uses its own Dalvik Virtual Machine (See Figure 3.1). This development process is identical to the developer as a standard Java platform. Application developers write their program in Java and compile java class files. However, instead of the class files being run on a J2ME virtual machine, the code is translated after compilation into a "Dex file" that can be run on the Dalvik machine. A tool called dx will convert and repack the class files in a Java .jar file into a single dex file with several shared constant pools. This is used to reduce the duplicated arguments and make the dex file more compact. The virtual machine itself is optimized to perform well on mobile devices with a slow CPU, limited memory, no operating system swap space and most importantly limited battery power. These constraints can be quite tight. For example, typically the total system memory will be approximately 64MB. High-level services will consume roughly 44MB. Then, the large system library will consume another 10MB. This leaves only 10MB for pure application code. The Dalvik VM is optimized for low memory compared to other standard VMs due to the following changes: the VM was slimmed down to use less space, Dalvik has no just-in-time compiler, the constant pool has been modified to use only 32-bit indexes to simplify the interpreter, and it uses its own bytecode, not Java bytecode.

File System: Android uses the YAFFS (Yet Another Flash File System), the first NAND optimized Linux flash file system. For mobile devices, hard disks are too large in size, too fragile and consume too much power to be useful. In contrast, flash memory provides fast read access time and better kinetic shock resistance than hard disks. There are fundamentally two different types of flash memory based on their construction technique: NOR and NAND. NOR is low density, offers slow writes and fast reads. NAND is low cost, high density and offers fast writes and slow reads. Embedded systems are increasingly using NAND flash for storage and NOR for code and execution. File systems for flash memory have to deal with these limitations in order to provide a robust file system.

Important limitations of NAND memory include block erasure and memory wear. Block erasure means that when erasing any memory the whole block must be erased. NAND can be randomly accessed on a page basis during programming, but cannot offer arbitrary random access rewrite or erase during normal operation.

To overcome this limitation, memory segments are marked to be removed or “dirty”. When the entire block is dirty, then it can be erased.

Memory wear means that there is limited number of erase-write cycles in the flash memory and at the end of its lifetime the data integrity of storage will deteriorate. Wear levelling techniques are used to uniformly use whole sections and to optimize the total lifetime of the device. Bad block management (BBM) is also used to perform write verification and remapping to spare sectors in case of write failure.

YAFFS was developed by Toby Churchill Ltd as a reliable filing system with fast boot time for their flash memory devices. They initially tried to modify existing flash file systems (used mainly for NOR) to add NAND support, but it turned out that the slow boot time and RAM consumption of existing flash file systems was unacceptable.

Furthermore, there are too many fundamental differences between NOR and NAND to make performance optimal. For instance, since erasing NOR is much longer than for NAND, garbage collection methodologies for NOR are not suitable for NAND. This led them to develop a different flash file system especially for NAND according to its features and limitations to optimize performance and ensure robustness. Upon completion YAFFS performed better than existing flash file systems and can still be used with NOR flash even though it was specifically designed for NAND.

YAFFS is the first flash file system specifically designed for NAND flash. It is highly portable and has been used on Linux, WinCE, pSOS, eCos, ThreadX and various special purpose operating systems. Compared to general purpose disk file systems, flash file systems eliminate seeking times, but they have to deal with lifetime limitations and error correction.
Mobile devices using YAFFS can usually tolerate far less errors in file system because the system can easily crash due to a corrupted file system. To avoid this problem, YAFFS provides bad block management and error correction to maintain data integrity in NAND flash and achieve a fast robust file system. The schema of YAFFS can be seen in Figure 3.2

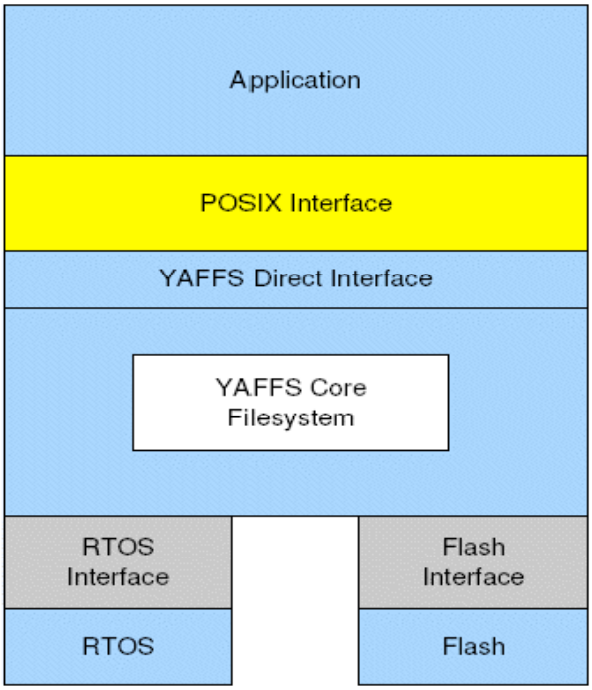


Figure 3.2: YAFFS embedded structure

YAFFS includes the following features: Journaling, Garbage collection, Lower memory requirement, Flexibility, Portability, Robustness, and POSIX Support. One can conceptually treat YAFFS as an improved version of flash file system since it outperforms previous flash file system in many ways.

Power Management: Power management in operating systems is necessary due to the ever increasing power demand of modern desktop computers and especially laptops. In order to reduce wasted power, multiple hardware power saving features are employed by Linux such as clock gating, voltage scaling, activating sleep modes and disabling memory cache. Each of these features reduces the system's power consumption at the expense of latency and/or performance. These tradeoffs on a Linux system are managed by either Advanced Power Management (APM) or Advanced Configuration and Power Interface (ACPI).

APM is an older, simpler, BIOS based power management subsystem, which is still used on older systems. Newer systems use ACPI based management instead.
ACPI is more operating-system centric than APM and also offers more features such as a tree structure for powering down devices so that subsystem components are not turned off before the subsystem itself.

In contrast with a standard Linux system, Android does not use APM, nor ACPI for power management. Android instead has its own Linux power extension, Power Manager instead. The core power driver (Shown at Figure 3.3 as "Power") was added to the Linux kernel in order to facilitate this functionality. This module provides low level drivers in order to control the peripherals supported by the Power Manager. These peripherals currently include: screen display and backlight, keyboard backlight and button backlight. Each peripheral's power is controlled through the use of Wake-Locks. These locks are requested through the API whenever an application requires one of the managed peripherals to remain powered on (Each lock setting shown in Table 3.1). If no wake lock exists which "locks" the device, then it is powered off to conserve battery life. In the case of multiple power settings the transition is managed through the use of delays based on system activity. A sample of this behaviour is shown in Figure 3.4 for the screen backlight. In addition to Wake-Locks the Power Manager also monitors the battery life and status of the device. This service coordinates with the power circuitry charging in the battery and also powers down the system when the battery reaches a critical threshold.

Table 3.1: Wake Lock Settings

Flag Value	CPU	Screen	Keyboard
PARTIAL_WAKE_LOCK	On*	Off	Off
SCREEN_DIM_WAKE_LOCK	On	Dim	Off
SCREEN_BRIGHT_WAKE_LOCK	On	Bright	Off
FULL_WAKE_LOCK	On	Bright	Bright

**If you hold a partial wakelock, the CPU will continue to run, irrespective of any timers and even after the user presses the power button. In all other wakelocks, the CPU will run, but the user can still put the device to sleep using the power button.*

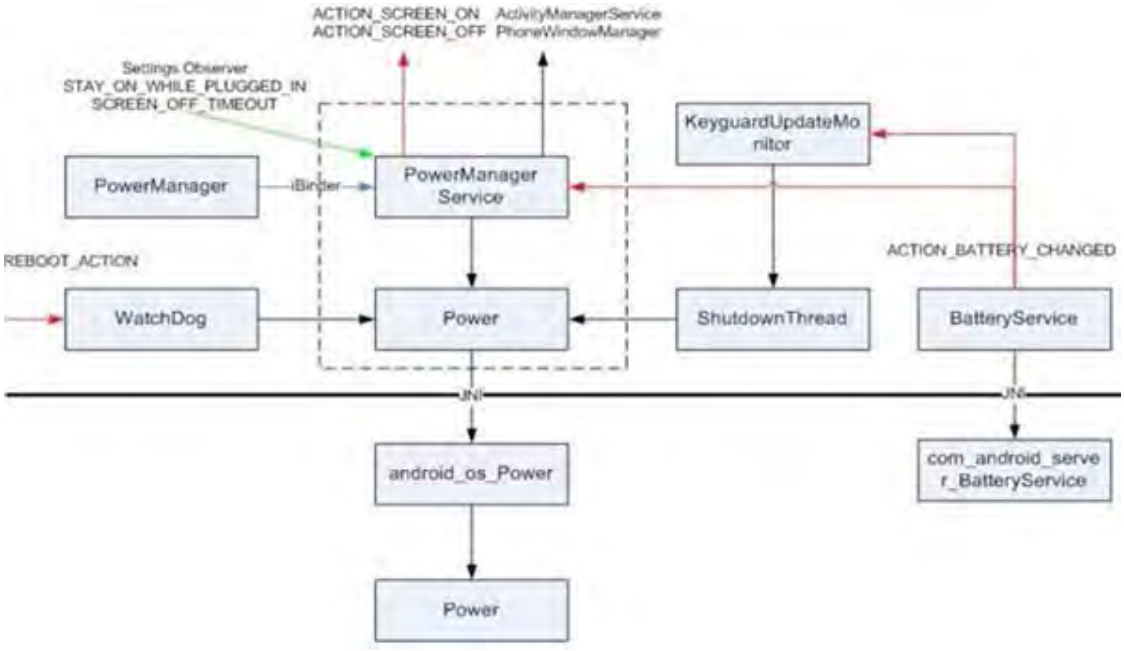


Figure 3.3: Android Power Management

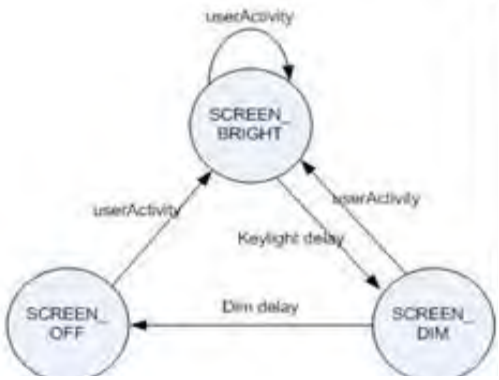


Figure 3.4: Power Management State Machine (Screen Brightness)

The following general topics were focused to clarify the design details of Android: architecture, kernel design, standard C Library, Java virtual machine, file system and power management. In order to understand how Android overcomes the hardware limitations of a mobile device as a light-weight operating system, it was depicted and summarized the key features of Android with a reasonable level-of-details in each topic. [10]

3.2 Symbian OS

The creation of Symbian OS can be traced back to a talented team of software developers at a company called Psion, an early pioneer in the handheld computer market. After successive generations of software for Psion's handheld devices, the team created an object-oriented operating system called EPOC, which was designed specifically for the unique requirements of mobile computing devices. Psion realized that there was a need for a mobile OS that could be licensed to other manufacturers for use in their mobile products, and that their EPOC operating system was well suited for this. At the time, the mobile phone industry was looking for a general operating system suitable for mobile phones and was interested in using EPOC. In June 1998, the software team stepped out on its own with the EPOC operating system and Symbian was born. Symbian was formed as a joint venture owned by other major mobile phone manufacturers as well as Psion, with the primary goal of licensing the EPOC operating system and improving it.

Fast forward to today, and we find that Symbian's operating system is a major player in the smartphone marketplace, residing in the majority of today's smartphone devices. Symbian is jointly owned by Nokia, Panasonic, Psion, Samsung, Siemens and Sony Ericsson which, together, represent a major portion of the mobile phone industry.

Symbian OS Overview: Symbian OS was designed from the ground up for mobile communications devices. While some competing operating systems (such as Microsoft's Smartphone OS) evolved from operating systems written for larger, more resource-laden systems, Symbian OS approached it from the other direction. Symbian's earlier versions (known as EPOC) would run on devices with as little as 2MB of memory.

Symbian OS is a multitasking operating system with features that include a file system, a graphical user interface framework, multimedia support, a TCP/IP stack and libraries for all the communication features found on smartphones.

Symbian OS has software development kits available for third-party application development. Also, the hardware layers of the operating system are abstracted, so that phone manufacturers can port the OS to the specific requirements of their phone. [11]

Symbian OS has a design that is more modular than many other operating systems. So, for example, disk services are in the main performed by the file server, and screen and user input services by the window server.

However, there is one element as the heart of the operating system – the element that is responsible for memory management, task management and task scheduling. That element is of course the kernel, EKA2.

There are many different flavours of operating system in the world, so to apply some adjectives to Symbian OS, and EKA2 in particular: Symbian OS and EKA2 are modular. As already said, operating system functionality is provided in separate building blocks, not one monolithic unit. Furthermore, EKA2 is modular too. There is no concept of multiple logins to a Symbian OS smartphone, unlike Windows, Mac OS X, UNIX or traditional mainframe operating systems.

EKA2 is multi-tasking. It switches CPU time between multiple threads, giving the user of the mobile phone the impression that multiple applications are running at the same time. EKA2 is a pre-emptively multi-tasking OS. EKA2 does not rely on one thread to relinquish CPU time to another, but reschedules threads perforce, from a timer tick. EKA2 is a priority-based multi-tasking OS with priority inheritance. EKA2 allocates CPU time based on a thread's priority and minimizes the delays to a high-priority thread when a low-priority thread holds a mutex it needs. EKA2 is real-time. Its services are (mostly) bounded, that is it completes them in a known amount of time. EKA2 can be a ROM-based OS. EKA2 is suitable for open but resource-constrained environments. It is designed for mobile phones, and so it needs less of key resources such as memory, power and hard disk than open desktop operating systems such as Windows or Linux.

The kernel is responsible for two key resources on a device: the CPU and the memory.
To isolate the kernel from different memory hardware designs, this interaction is encapsulated in a distinct architectural unit called the ‘‘memory model’’. With the different memory models provided with EKA2 one can find out how they use the memory address space (the memory map) and their contribution to overall system behaviour.

The memory model: At the application level – and to a large extent when writing kernel side software – the main use of memory is for allocation from the free store using operator new or malloc. The kernel has the following responsibilities related to memory management: Management of the physical memory resources: RAM, MMU(Memory Management Unit) and caches, Allocation of virtual and physical memory, Per-process address space management, Process isolation and kernel memory protection, and the memory aspects of the software loader. As well as providing these essential services, it is needed to ensure that the design of the memory model does not impose hard or low limits on the operating system. The provision of efficient services to carry out these responsibilities is dependent on the memory architecture in the hardware. In particular, a design that is fast and small for some hardware may prove to be too slow or require too much memory if used on another. One of the aims of EKA2 was to be readily portable to new hardware, including new MMU and memory architectures.

F32 (File Allocation Table (FAT) 32) system architecture: The entire file server system consists of the (shaded) components displayed in Figure 3.5. The file server, like any other server in Symbian OS, uses the client/server framework. It receives and processes file-related requests from multiple clients. The file server runs in its own process and uses multiple threads to handle the requests from clients efficiently. Clients link to the F32 client-side library (EFSRV.DLL). The file server executable, EFILE.EXE, contains two servers – the file server itself and the loader server, which loads executables and libraries.

Because of the different characteristics of the various types of disk that Symbian OS supports, there are a number of different media formats. For example, removable disks are FAT formatted to be compatible with other operating systems, and the ROM drive uses a format scheme which is efficient for read operation, but which wouldn’t be suitable if writes to the drive were required. In general, the file server does not concern itself with the detail of each file system; instead the different media formats are implemented as separate file systems, components that are ‘‘plugged into’’ the file server. The exception to this is the ROM file system, which is built into the file server.

File system components are polymorphic DLLs that have the file extension ‘‘.FSY’’.
These DLLs are dynamically loaded and registered with the file server, normally at system boot time. Figure 3.5 shows a file server configuration with two file systems loaded, ELOCAL.FSY and ELFFS.FSY.

Before a particular drive can be accessed, it must have a file system associated with it, whereupon it can be said that the drive is mounted. Again, the file server generally carries out this process at system boot time, once the file systems have been loaded.

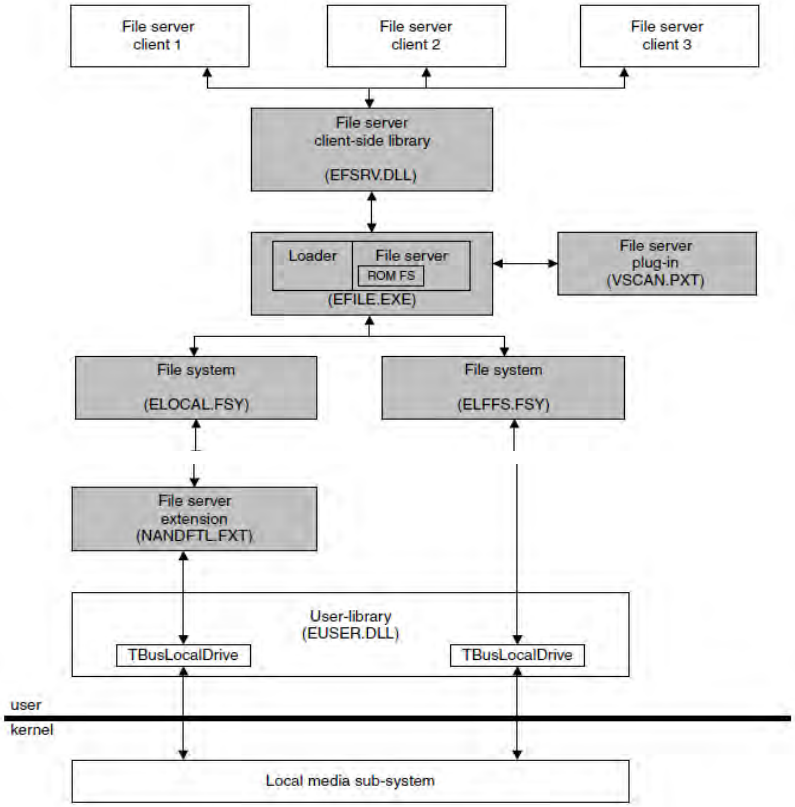


Figure 3.5: F32 system architecture

The file systems gain access to the mobile phone’s internal and removable disks via a set of device drivers known as the local media sub-system. These drivers are split into a logical device driver layer – the local media LDD (ELOCD.LDD) and a physical device driver layer.

The physical device drivers are called media drivers. The user-side interface to the local media sub-system is provided by the class TBusLocalDrive whose methods are exported from the user library (EUSER.DLL). The main functions it provides are those to read, write and format regions of each drive's memory area.

Again, the ROM drive is an exception, as it is not accessed through the local media sub-system. Instead, the bootstrap maps this memory area to be user-readable, and the file-server accesses it directly. For other drives though, the TBusLocalDrive class provides the user side interface to the physical media. Often, when a file system is mounted on a particular drive, it will interface directly with the TBusLocalDrive instance for that drive. This is the case for the file system ELFFS.FSY.

Power Management: Mobile phones are battery-powered devices. In the majority of cases, the battery is the only available source of energy – the exception being the times when the mobile phone is being recharged. Even with today's most advanced developments, rechargeable batteries are still characterized by: the limited amount of power they can supply at any given time, the limited period for which they can supply electrical energy before exhaustion (the depletion of the active materials inside a cell must be replenished by recharging), and the hysteresis of the depletion–recharge cycle, which shortens the life-span of a battery.

So it is fair to say that the supply of power on a mobile device is quite constrained. The problem is compounded by the need to keep the size and weight of battery components as small as possible. At the same time, new features are being added that use more power, or users are operating phones in power-consuming states (such as gaming or audio/video playback) for longer periods of time.

Because of this, mobile phone hardware has gained new energy-saving features, which require software monitoring and control. It is the primary goal of the operating system's power management architecture to define and implement strategies to use energy efficiently, to extend the useful life-time of the batteries, to increase the period of time for which the device can be used between recharges and at the same time, to allow the use of services required by the user of the device at any given time and at an acceptable level.

Power management deals with the operational state of the mobile phone as perceived by its user. That perception is primarily based on the availability of the user interface. The device is seen to be operational when the user can interact with the UI. On the other hand, the device is perceived to be unavailable when the UI seems to be unavailable, for example when the display is off or is displaying a screen saving image. From this, it is clear that power management must be implemented at all levels of the operating system.

Symbian OS favours a distributed approach to power management, with components at different layers of the OS responsible both for managing their requirements on system power, and the impact of their actions on the availability of the phone. They achieve this in co-operation with other interdependent components, which can be at any level of the OS.

Power states: The kernel-level implementation of power management is responsible for managing the power state of hardware components such as the CPU and peripherals.
The kernel-side framework sees a hardware component's power state as Off, Standby, Retention, Idle, or Active.

Transitions between power states may be triggered by user actions, requests from the clients of the services provided by components, the need to save power, changes to the state of power resources used by the component, and so on. These states apply to the CPU and peripherals independently, so it is possible that, at a given time, different hardware components of a phone will be in different power states. For example, it is possible that the CPU might enter the idle state if it has no scheduled tasks and there are no anticipated events requiring its attention, while a peripheral has an outstanding request for servicing incoming data. [6]

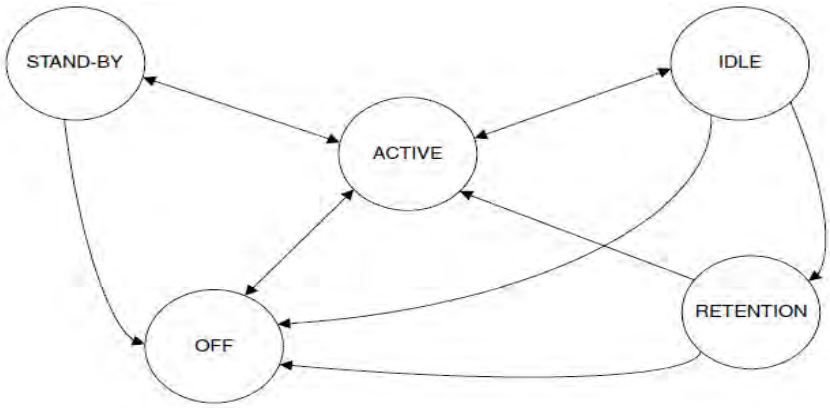


Figure 3.6: Typical CPU state transition diagram

Symbian OS Various Flavours: It is challenging to create an operating system that provides common core capabilities and a consistent programming environment across all smartphones – yet at the same time allow for manufacturers to differentiate their products. Smartphones come in many different shapes and sizes with varying screen sizes and user input capabilities; the user interface software needs to vary to fit these differences.

Symbian OS has a flexible architecture that allows for different user interfaces to exist on top of the core operating system functionality. Of course, it is not wise to be too flexible for two reasons: having too many different user interfaces inhibits code reuse among different devices and too much work is required to create a GUI from scratch for their smartphone.

So, to give the phone makers a starting point, Symbian created a few reference platforms, each packaging the Symbian OS core functionality along with a user interface that matched one of the basic smartphone form factors (screen size and input capability). These are Series 60, UIQ and Series 80. [11]

Symbian OS architecture

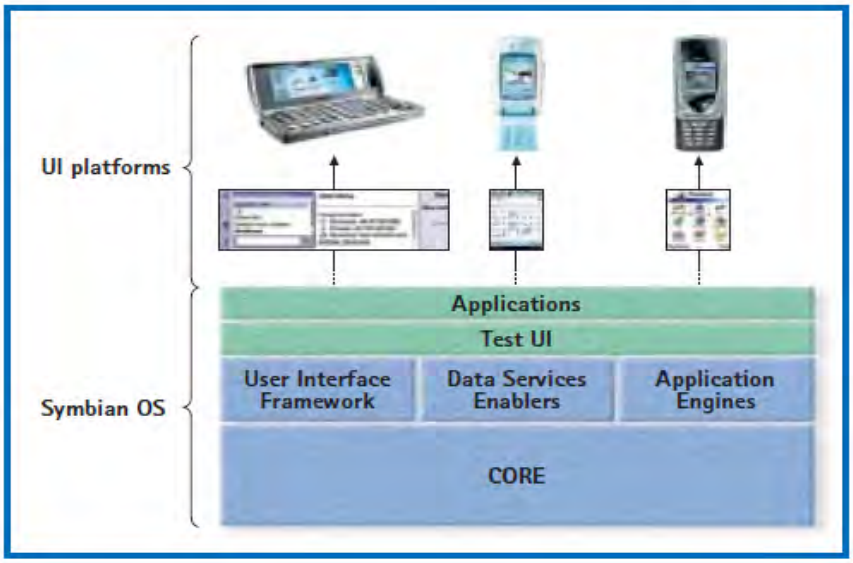


Figure 3.7: Symbian OS Architecture

Symbian OS architecture is designed to meet a number of requirements. It must be hardware independent so it can be used on a variety of phone types, it must be extendable so it can cope with future developments, and it must be open to all to develop for.

Architectural overview: Core – Symbian OS core is common to all devices, i.e. kernel, file server, memory management and device drivers. Above this core, components can be added or removed depending on the product requirements.

System Layer – the system layer provides communication and computing services such as TCP/IP, IMAP4, SMS and database management

Application engines – above the Systems Layer sit the Application engines, enabling software developers to create user interfaces to data.

User Interface Software – can be made or licensed by manufacturers (for example in the case of the Nokia Series 60 platform).

Applications – are slotted in above the User Interface.

Client Server Architecture, Event management, Object oriented design, Robust (Devices should not lose user data, crash or require rebooting) and dependable, Full multitasking are also basic symbian OS features of strength.

An open operating system: Symbian OS is "open" – it means Open to anyone to license, Open to anyone to develop applications, it is based on open standards, and Owned by the industry.

Writing applications for Symbian OS (Java): All Symbian OS devices have Java available on them. The higher end devices tend to have Personal Java and the more popular devices have MIDP Java. Programming in Java for Symbian OS is the same as programming for Java on any other OS. [12]

3.3 Blackberry OS

The Blackberry Operating System is a software platform developed by its manufacturer RIM

Its OS provides multi-tasking that maximizes use of the devices specialized platform including: trackball, trackpad and touchscreen.

Updated versions of the Blackberry OS are released regularly to support new BlackBerry Smartphones; latest OS version is OS 5.0. The current version of the OS allows complete wireless activation and synchronization with Exchange's email, calendar and other features.

Blackberry OS Architecture: For example if BlackBerry 9000 series have chosen, which runs v5.0 of the BlackBerry OS; this version of the BlackBerry OS has a Java based kernel, and utilizes ARM architecture with an Intel XScale processor. ARM is a Reduced Instruction Set Computer (RISC) type instruction set architecture. It uses 16 x 32-bit registers, 1 processor status register and load/store architecture.

Memory Management: Memory is divided into three sections: Application Memory (~128MB) -a dedicated memory space for application storage and overhead, Device Memory (~850MB) -for storing files and other media, and Memory Card (optional) -an optional method of file storage.

A common criticism of the Blackberry is that Device Memory cannot be allocated to supplement Application Memory. This is especially inconvenient as Application Memory handles all the overhead for running applications. If the device also has memory card storage, this makes the Device Memory redundant.

Also, the memory manager does not release memory after applications are closed, which can lead to a considerable slowdown of the device over time or prolonged use; this is a major drawback for a device that is primarily marketed at those in business.

Multi-Tasking: The Blackberry supports multitasking. It can thus run more than one application at a time. For example: while making a call, you can switch to the calendar or contacts application. These applications run in the background while carrying out current task, however, the more applications that are running, the more memory used by device [13]

Ease of Use: There's a very good reason why Blackberry is a household name and Windows Mobile is not: Its overall design and usability are intuitive and rock solid, so we give it high marks in these areas. Setting up and using e-mail, surfing the Web, and making phone calls were simple tasks. The icon-based Home screen is a breeze to follow, and the menus are well integrated with the function keys. Blackberry’s keyboard shortcuts are also great time-saving tools.

Web Browsing: Pages, such as CNN.com, loaded pleasingly fast-if less visually stunning - using AT&T's EDGE connection on our Blackberry Curve. It took about 12 seconds before we could start reading CNN.com's homepage on the Curve. On the other hand, Blackberry doesn't yet support Wi-Fi or HSDPA. If 3G speeds is needed, CDMA Blackberry device like the 8800 should be available, which is available from Sprint and Verizon Wireless.

Multimedia: Blackberry could use some improvement in this category, such as providing the ability to play Flash video files found online. The operating system can play various audio and video files, including MP3, AAC, and WMA for music and MP-4 and WMV for video. Blackberry has its Desktop Media Manager.

Third-Party Apps: Many third-party software developers build enterprise and consumer applications for Blackberry. The site of the leading mobile-content provider, Handango, had more than 950 software titles available for download for Blackberry. Blackberry is only now starting to become more multimedia-friendly for developers, with the addition of several games

Performance & Battery Life: Based on the experience using the Curve and other Blackberries, the OS is stable and runs smoothly. The Blackberry easily handled pretty much any task threw at it. It is able to navigate menus and browse Web pages, for example, quickly and easily. Battery life depends, of course, on usage and on the network the phone supports. Blackberry Curve battery life lasted for a few days of typical daily chores such as several short calls, checking e-mail, IMing, Web browsing, shooting photos, and listening to and watching a few songs and videos. Blackberry has also evolved, especially on the multimedia front, but ultimately it is preferred because it doesn't feel like it's trying to do so much. The OS is snappier at opening and closing applications and surfing the Web, and it has lots of helpful shortcuts to make already intuitive software even more useful. [14]

3.4 Windows OS

Windows Mobile is a mobile operating system developed by Microsoft that was for use in smartphones and mobile devices, but is being phased out to specialized markets. The current version is called "Windows Mobile 6.5". It is based on the Windows CE 5.2 kernel, and features a suite of basic applications developed with the Microsoft Windows API. It is designed to be somewhat similar to desktop versions of Windows, feature-wise and aesthetically.

Additionally, third-party software development is available for Windows Mobile, and software applications can be purchased via the Windows Marketplace for Mobile.

Originally appearing as the Pocket PC 2000 operating system, most Windows Mobile devices come with a stylus pen, which is used to enter commands by tapping it on the screen. Microsoft announced a completely new phone platform, Windows Phone 7, at the Mobile World Congress in Barcelona on February 15, 2010. Phones running Windows Mobile 6.x will not be upgradeable to version 7.

Windows Mobile's share of the smartphone market has fallen year-on-year, decreasing 20% in 2009. It is the 5th most popular smartphone operating system, with a 5% share of the worldwide smartphone market (after Symbian, Blackberry OS, Android and iPhone). In the United States, it is the 3rd most popular smartphone operating system for business use (after Blackberry OS and iPhone), with a 24% share among enterprise users. Microsoft is phasing out Windows Mobile to specialized markets, such as rugged devices, and focusing on its new mobile platform, Windows Phone 7.

There are three versions of Windows Mobile for various hardware devices: Windows Mobile Professional runs on (smartphones) with touchscreens ,Windows Mobile Standard runs on phones with regular screens and Windows Mobile Classic which runs on 'Windows Mobile Classic devices' (Pocket PCs).

Windows Mobile 6, formerly codenamed "Crossbow", was released on February 12, 2007 at the 3GSM World Congress 2007. Windows Mobile 6 is powered by Windows CE 5.0 (version 5.2) and is strongly linked to Windows Live and Exchange 2007 products. Windows Mobile 6 Standard was first offered on the Orange's SPV E650, while Windows Mobile 6 Professional was first offered on the O2's Xda Terra.

Aesthetically, Windows Mobile 6 was meant to be similar in design to the newly released Windows Vista. Functionally, it works much like Windows Mobile 5, but with much better stability. [15]

Taking the reins from Windows Mobile 5, Windows Mobile 6 isn't a complete overhaul of the OS; instead, it offers a number of useful enhancements that makes performing tasks easier and puts more powerful tools into the hands of mobile professionals.

It particularly impresses with the new e-mail search function, Mobile Office additions, and Windows Live integration, but Microsoft could have done a lot more. For example, multimedia improvements are practically nonexistent and the user interface is still kludge, requiring numerous steps to complete a simple task. Also, some of the enhanced functionality to Outlook and calendaring require using Exchange Server 2007. Despite these flaws, the new improvements make Windows Mobile 6 worth the upgrade.

Windows Mobile 5 users won't be in for any major surprises when they see Windows Mobile 6, as the interface largely remains the same as before. Windows Mobile 6 does have more of a Vista look with its similar colour scheme and bubbly, eye-pleasing icons. Along the top of the Today screen, there are shortcuts to most recently used apps, but the icons are slightly larger. Below that, important information as time, date, upcoming appointments, messages, and so forth are found. Of course, one can customise the background image, colour scheme and backlight time.

One of the biggest complaints about Windows Mobile devices, especially when compared to Palm, is the number of steps it takes to perform a simple task, such as closing out of a program. This is still pretty much true of Windows Mobile 6, but Microsoft has taken some steps to ease the pain. For example, the company has added nine new e-mail shortcuts so it is easy to reply, delete, move messages, and more. While this is a step in the right direction, there is still plenty of room for improvement.

Window Mobile 6 really doesn't offer any mind-blowing new features, but rather, it includes some nice refinements that make the devices easier to use as well as act more like PC.
Starting with some of the basics, call history is now sorted to the appropriate contact page. Though it doesn’t seem a big deal, it's actually quite convenient to easily see when was a received and made call to specific person, the time of the call, the duration, and so forth. Also, the new OS provides a quick Send Text Message shortcut. The Calendar application is also more users friendly.

E-mail is a lot smarter on Windows Mobile 6. Searching for e-mails is no longer an unpleasant task, thanks to a new search function similar to the Smart Dial feature on Windows Mobile 5 devices. If you have a Hotmail/Windows Live e-mail account, you can easily access those messages with Windows Live for Mobile.

The big news here is that Windows Mobile 6 Standard Edition (formerly Smartphone Edition) now has the full Microsoft Office Mobile Suite. [16]

Windows Mobile Professional Power Management: Windows Mobile Professional has four power modes. In the full powered state the device's screen and backlight are on. If the power button was not pressed or if there is no interaction with the device, after so much time it goes into the unattended state. In this state the device is still running but the screen and backlight are powered down. The user would look at this is being an off state but the device is still awake. After about 15 seconds the device will go from the unattended state to the suspended state. In the suspended state all of the programs that had been running are still in memory but they make no progress since the CPU is in a halted state. The final state is the true off state. [17]

Windows Mobile Persistent Memory: One of the most talked aspects of the Windows Mobile 5 OS is its redesigned memory paradigm called "Persistent Memory Storage." This section explains the memory management scheme. In all versions of the Pocket PC and/or Windows Mobile operating systems before Windows Mobile 5, Pocket PC memory was managed in the way illustrated below. RAM and ROM chips worked together to store data and run applications. The operating system was stored in the ROM and could not be changed; in this way, the unchanged OS could be restored in the event of an error. The RAM memory chip pulled double duty. In addition to storing active programs and files, just like the RAM in PC, it also stored all data, applications and settings that entered from the time it first turned on, like PC's hard drive.

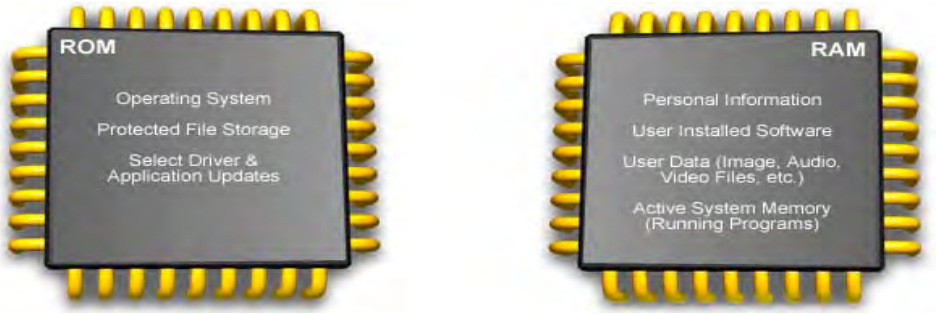


Figure 3.8 Previous Memory Scheme

Persistent Memory Storage, the Windows Mobile 5 memory scheme, manages memory more like desktop or laptop PC. The ROM chip stores all data (like PC's hard drive), and the RAM is dedicated to active memory (like PC's RAM).

Figure 3.9: Persistent Memory Storage: WM5

Persistent Memory Storage has several advantages over the previous memory scheme. First, because all information is stored in the ROM, losing power does not clear Pocket PC's memory. Second, since all data is saved in the non-volatile ROM chip, the RAM chip has no need for constant power from the battery, so the battery is not always being drained. Third, because the full capacity of the RAM is free for active system memory, programs should run faster and more applications will be able to run at the same time.

Persistent Memory Storage is also built into Windows Mobile 6. [18]

4. DESIGN AND IMPLEMENTATION OF BENCHMARK

4.1 Software Environments

Android Development Environment

This section describes how to install the Android SDK and set up the development environment for the first time. If one is currently using the Android 1.6 SDK or later and want to update to the latest tools or platforms, it does not need to install a new SDK. Instead, simply update the individual components in the SDK using the Android SDK and AVD Manager tool. If it is Android 1.5 SDK or earlier, new SDK should be installed.

Step 1 Preparing Development Computer: Before getting started with the Android SDK, take a moment to confirm that the development computer meets the System Requirements. In particular, install the JDK before continuing, if it's not already installed.
If it is needed to develop in Eclipse with the Android Development Tools (ADT) Plug-in make sure that a suitable version of Eclipse is installed on the computer (3.4 or newer is recommended). For this work Eclipse 3.5 was installed.

Step 2 Downloading the SDK Starter Package: The first step in setting up the environment for developing Android applications was downloading the Android SDK starter package. The starter package is not a full development environment — it includes only the core SDK Tools, which was used to download the rest of the SDK components.
After downloading, unpack the Android SDK archive to a safe location on the machine. By default, the SDK files are unpacked into a directory named android-sdk-<machine-platform>. Optionally, it is needed to add the location of the SDK's primary tools directory to system PATH. The primary tools/ directory are located at the root of the SDK folder. Adding tools to the path lets to run Android Debug Bridge (adb) and other command line tools without supplying the full path to the tools directory.
The next section describes how to install the Android Development Tools (ADT) plug-in and set up Eclipse. One can develop Android applications in an IDE of the choice and then compile, debug and deploy using the tools included in the SDK.

Step 3 Installing the ADT Plug-in for Eclipse: Android offers a custom plug-in for the Eclipse IDE, called Android Development Tools (ADT) that is designed to give a powerful, integrated environment in which to build Android applications.

It extends the capabilities of Eclipse to quickly set up new Android projects, create an application UI, add components based on the Android Framework API, debug applications using the Android SDK tools, and even export signed (or unsigned) APKs in order to distribute application. In general, developing in Eclipse with ADT is a highly recommended approach and is the fastest way to get started with Android. Eclipse IDE was used with ADT in this work.

Step 4 Adding Android Platforms and Other Components: The last step in setting up SDK is using a tool included in the SDK starter package — the Android SDK and AVD Manager — to download essential components into the development environment. The SDK uses a modular structure that separates the major parts of the SDK — Android platform versions, add-ons, tools, samples, and the API documentation — into a set of separately installable components. The SDK starter package, which has been already downloaded, includes only a single component: the latest version of the SDK Tools. To develop any Android application, it is needed to download at least one Android platform into the environment. The SDK repository offers these types of components: SDK Tools, Android platforms, SDK Add-Ons USB Driver for Windows, Samples, and Documentation. Android 2.2 platform was installed for this work.

The SDK repository contains a range of components that one can download. Anyone can determine which components are needed, based on whether a basic (but functional) development environment or a recommended or full development environment is wanted.

Step 5 Exploring the SDK: Once the SDK was installed and downloaded the platforms, documentation, and add-ons, open the SDK directory and take a look at what's inside.

Once completed installation, it is ready to begin developing applications. [19]

Symbian Development Environment

Before installing the SDK, check the minimum hardware and software requirements. With these minimum requirements, one can run the SDK as a standalone application. If it is planed to use the SDK with an integrated development environment (IDE), the requirements mandated by the IDE in question should be seen.
Software Requirements are Active Perl version 5.6.1, Java Runtime version 1.5.0, and ARM RVCT compiler 2.2 build 593 or newer is supported for ARMV5 compilation.

The following operating systems are supported: Microsoft Windows XP Professional SP2, Microsoft Windows Vista Business. All installations must be performed using an administrator account. Otherwise, some environment variables may not be set correctly. The SDK must be installed in a path that does not contain any white space characters. The SDK, IDE and project files must be located on the same logical drive as build tools to work correctly in all situations.

Before installing the SDK one should have installed and configured: Perl, Java, and peripherals that are going to use. ActivePerl 5.6.1 is required and must be installed before installing this SDK. ActivePerl is downloaded from the ActivePerl pages. Java Run-Time (JRE) 1.5.0 is required to use emulator Preferences and other Java components. Java Run-time can be downloaded from Sun Developer Network Downloads. The systems used in this work were MS Vista Home premium and ActivePerl 5.10.1.
Once downloaded the installation package (.zip file) on the PC, installing the S60 SDK takes place through the SDK Installer, that is, the Install Shield Wizard. The following are steps needed to install the SDK.

1. Start the SDK installation by running the installation executable setup.exe located in the SDK delivery ZIP file. If S60 5th Edition SDK v0.9 is installed, the installation is aborted, prompting the user to uninstall the S60 5th Edition SDK v0.9: S60 5th Edition SDK v1.0 cannot be installed until S60 5th Edition SDK v0.9 is uninstalled.
2. Click Next button to continue. The License Agreement dialog is displayed: Read the license agreement carefully.
3. After reading (and accepting) the license agreement, click the I accept the terms of the licensee agreement radio button and click Next

The SDK installer provides the following installation options: Typical this is the recommended installation option, Compact, and Custom. Select the type of installation that is required by clicking the appropriate radio button.

To install the SDK not in the proposed default directory (C:\S60\devices) browse the installation directory by clicking the Browse button.

Once selected the installation type and (possibly) defined the installation directory, click Next.

4. If the Custom installation option is selected, the Select Features dialog is displayed:

The options are: CPP (C++), MIDP, and COMMON. Click the features to include in the SDK installation and click Next.

5. The Choose Destination Location dialog is displayed: Define Eclipse installation directory by clicking Browse and then Next. C++ users may skip this stage by clicking NEXT button, as JAVA documentation is not relevant for C++ developers
6. The Ready to Copy Files dialog is displayed. Click Next to start the SDK installation. The SDK installation status is displayed in the Setup Status dialog.
7. The Start Copying Files dialog is displayed. Click Next to start the SDK installation.

The SDK installation status is displayed in the Setup Status dialog.

8. If other SDKs were installed on the PC, the Install Shield Wizard will prompt to select one of them as the default SDK. Select the appropriate SDK in the dialog and click Next.
9. If the CSL ARM Tool chain was not installed on the PC, the following dialog will appear, prompting to install it: As the CSL ARM Tool chain contains for example the GCCE compiler.
10. To complete the installation, click Finish in the Installation complete dialog.

The SDK is now fully installed on the PC. It can be verified through the Windows Start menu by, for example, opening the SDK Help by selecting Start > All Programs > S60 Developer Tools > 5th Edition SDK > v1.0> SDK Documentation. [20]

Blackberry Development Environment

This section shows which tools are needed to start developing applications for the Blackberry Platform, where to find those tools and how to install them. The Blackberry platform is Java based, and there are number of different tools that can be used to develop applications.

This section focus on Eclipse, as it is one of the most widely used tools in the industry.

The best thing is all the tools required to develop applications for Blackberry are absolutely free.

Basic System requirements are Microsoft Windows Vista, or Windows XP, Java SE Development Kit (JDK) version 5 or version 6- Version 6 is required if Blackberry MDS-CS is used for debugging, Blackberry JDE Plug-in for Eclipse with an existing installation of the Eclipse IDE and Blackberry Java Development Environments (JDEs).

First download all the necessary files: the JDK, the Eclipse IDE, the Blackberry tools.

From the Blackberry tools download the Blackberry Plug-in for Eclipse, as well as the Blackberry JDEs. Downloading all available JDEs as that will enable to test applications on most current Blackberry devices is recommended. The plug- in and at least one JDE is required.

It is important to install the JDK first and Eclipse second before starting installing the Blackberry tools. [19] These two soft wares were previously installed in the laptop. Then Blackberry JDE v5.0 and Blackberry Plug-in for Eclipse was installed for this thesis work.

Windows Development Environment

The Windows Mobile 6 SDKs add documentation, sample code, header and library files, emulator images and tools to Visual Studio to build applications for Windows Mobile 6.

Windows Mobile 6 SDK comes in three forms Windows Mobile Standard, Windows Mobile Classic and Windows Mobile Professional.

The following emulator images are included in the SDKs: Windows Mobile 6 Standard SDK includes Windows Mobile 6 Standard, and Windows Mobile 6 Standard Landscape QVGA. Windows Mobile 6 Professional SDK includes Windows Mobile 6 Classic, Windows Mobile 6 Professional ,Windows Mobile 6 Professional Square, Windows Mobile 6 Professional Square QVGA ,Windows Mobile 6 Professional Square VGA, and Windows Mobile 6 Professional VGA.

In System Requirements supported Operating Systems are Windows Server 2003 Service Pack2, Windows Vista, and Windows XP Service Pack 2. The OS supported in this work was Windows Vista. For this OS the requirements was Microsoft Visual Studio 2008 Professional Edition and above or Microsoft Visual Studio 2005, Standard Edition or above (Express Editions are not supported), Microsoft.

NET Compact Framework v2 SP2 and for synchronizing data, the Windows Mobile Device Centre is required. Microsoft Visual Studio 2008 Professional was installed for this work.
"Visual C ++ Smart Device Programmability", which is an option of Microsoft Visual Studio Setup, must be selected and installed during Visual Studio installation.
Windows Mobile 6 Professional SDK Refresh does not contain the Windows Mobile Standard SDK Refresh and you will need to download both to target both platforms.
Windows Mobile 6 Professional SDK Refresh contains Windows Mobile 6 Professional emulator images and one Windows Mobile 6 Classic emulator image. [22]

4.2 Design of Benchmark Applications

In this section Design of Benchmark application of each system is discussed in detail. The full code for each application is attached in the Appendix.

Android Benchmark application

As described earlier the language used for designing this Benchmark application is Java, which is specific to android. How to write, what is written and how to run the application is discussed in this section. The required software environments explained in the previous section were installed well. Then the steps followed is launching Eclipse, click File-New-Android Project- then fill required fields like project name, Application name(optional), package name , create Activity and fill Minimum SDK version then click on Finish to end the creation of new Android project. In the project created src, rec, and AndroidManifest.xml are components found. The step followed next was creating class. Being in the package New Class was created.

As standard java programming; writing code starts with the header and continues to the body of the code is used here. In the application a media was started to play when a button was clicked. For this to work there is OnClickListener interface that is implemented. The class declaration line was as shown here.

public class BenchMarkAndroid extends Activity implements View.OnClickListener

4 Design and Implementation of Benchmark

Before the media starts playing initial information of the operating system features were recorded. Recorded data are current Battery percent, current Free RAM memory, and starting time. Battery percent is fetched using the method BroadcastReceiver().[23] The code fragment is as shown below.

```
private BroadcastReceiver mBatteryInfoReceiver = new BroadcastReceiver() {

public void onReceive(Context context, Intent intent) {

if(Intent.FLAG_FROM_BACKGROUND = =4){

String action = intent.getAction();

int level = intent.getIntExtra("level", 0);

int scale = intent.getIntExtra("scale", 100);

tvBatteryLevel.append ("\n Battery level: " + String.valueOf (level * 100 / scale) + "%");}}}
```

Free RAM memory is also fetched using the following code fragment [24]

```
ActivityManager.MemoryInfo mInfo = new ActivityManager.MemoryInfo ();

actvityManager.getMemoryInfo( mInfo );

tvBatteryLevel.append("Memory usage at the start of application \n ");

tvBatteryLevel.append("Available memory is "+

String.valueOf(mInfo.availMem/1048576L)+ "MB");
```

For getting system current time simply System.currentTimeMillis () method is used.
At the middle of playing media the metrics measured were read once more. To get the middle point of playing audio file the getCurrentPosition method was instantiated as follows

```
if(mp.getCurrentPosition()= =(mp.getDuration()/2))
```

At the end of playing the media the metrics were read for the third time. Completion of the playing media is known by using the OnCompletionListener() method called as below.

```
mp.setOnCompletionListener(new OnCompletionListener())
```

To test Multitasking ability of the operating system second job that is done in parallel with playing audio was given. This work is downloading image file from the internet. To initialize the second work another button was developed. With event listener OnLongClickListener the following method is called when the button is pressed for long time.

public boolean onLongClick(View v2)

The second class was invoked to do works in parallel with the first class. In the second class method OpenHttpConnection(URL) played major role in connecting emulator/device to the internet. Once it is connected an image is downloaded as Bitmap data type. Both HttpDownload and BenchMarkAndroid classes resides in the same package. [25]

The data needed to run the application, i.e. media that is going to be played and pictures of the button were saved in the resource folder (res). Inside the res folder a new folder “drawable” was created and the picture used for buttons was saved there.

Also in the res folder a new folder “raw” was created and the audio file that was played was saved. Layout folder inside the res folder contains .xml files that determine the layout of the output pages. In the project there are two xml files for both classes developed.

In the Manifest file permission needed for the project was allowed and each activity was declared in side this file. In this project internet connection was permitted in side this file.

To run the application, the procedures were selecting the project and right click and click Run As-Android Application. In this time the simulation starts to launch and output data were being displayed.

The flow chart shown below in Figure 4.1 describes Android Benchmark application internal components. As it is seen from the figure basic works done were Display measured metrics at different points, playing audio file and downloading image file. The works are done by checking conditions such as play button pressed, if middle of play and end of play are sensed.

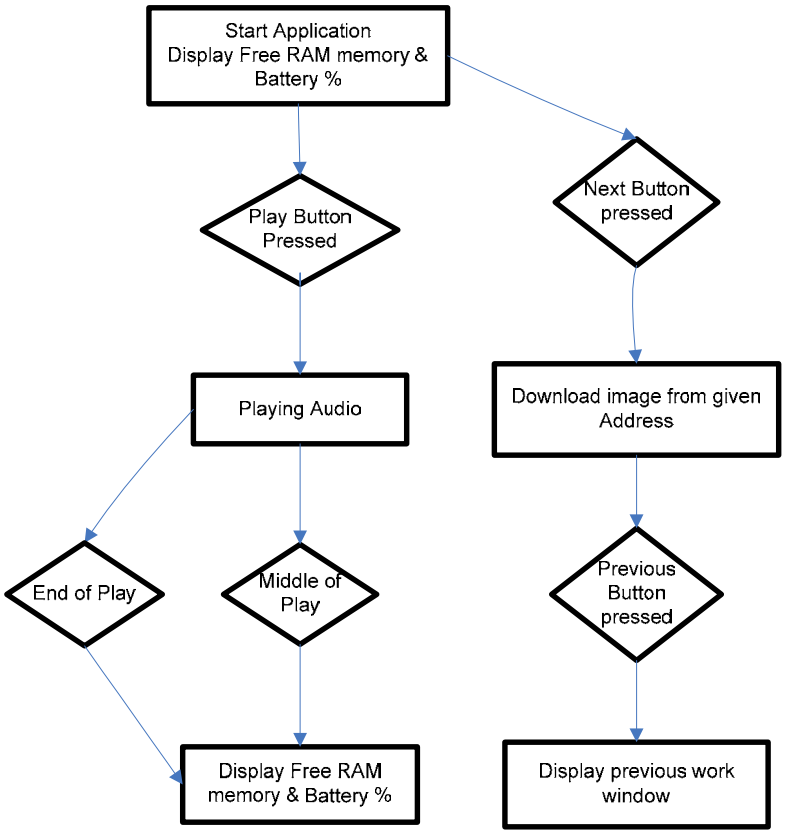


Figure 4.1: Android Benchmark Flow chart

Symbian Benchmark application

The application has been developed in eclipse environment with symbian platform SDK is installed. The language used for development is java Micro edition.

The steps followed to create new project was click File-New-Project-J2ME-J2ME Midlet Suite then Click Next and fill project name in the space provided and Click Next. The next step was to decide the software that is going to execute the project and choose either emulator or real device in which the project is run on. Group and Device for software environment selection and Emulator or real device selection respectively was decided. The last step was to finish the wizard by clicking on Finish button.

In the project created there are folders as src and res and also library components. The source code and the audio files were put in the src folder. The step followed was creating a class; being in the package New Class was created.

To run the application the following steps was performed, Right Click on the application class- choose Run As and then Emulated J2ME Midlet, by doing this the emulator started to launch the application.

In the application there were two Forms for displaying the two parallel works separately.

```
Public Form form1, form2
```

In the forms there was commands used to instantiate works. The following commands were defined in the application.

```
private Command exitCommand,viewimg,cmBack,play;
```

```
Each command has attached action that was done when the command was selected
```

```
if (command == exitCommand) { exitMIDlet(); }
```

```
if (command == viewimg) { imagedownload(); }
```

```
if (command == play) { playmedia();}
```

```
if (command == cmBack)
```

```
{display = Display.getDisplay(this);
```

```
display.setCurrent(form2);  }
```

Except the cmBack command all the commands were attached with the first Form. In the First Form viewimg command is used to start the second work (downloading image file).

When the application starts to run, data measured by the Benchmark was read in the same manner as in the application described in Android platform. In the case of symbian applications they start running from the method startApp ().

Then using the getProperty method free RAM memory and Battery level was read from the device as shown in the code fragment below.

```
public void startApp() {
```

platform = System.getProperty("com.nokia.memoryramfree"); [26]

memo=Long.parseLong(platform)/1048576L;

batt=System.getProperty("com.nokia.mid.batterylevel"); }

When play command is selected from the Form the audio starts to play and when it was reached at the middle of the play data measured by the benchmark was displayed for the second time. For the third time when media play reached End of media metrics were read from the device and added to the Form. The condition is shown below.

if (event == PlayerListener.END_OF_MEDIA)

To test multitasking ability of the operating system the "viewimg" command should be selected. In the time when "viewimg" command was selected a function is called that tries to connect internet and download image from the given URL address. The function's fragment code looks like this: [27]

private Image getImage(String url) throws IOException

{ContentConnection connection = (ContentConnection) Connector.open(url);

DataInputStream iStrm = connection.openDataInputStream(); }

The image was read as byte array and will be changed to image format later as shown in the code here.

byte imageData[]=new byte[length];

ByteArrayOutputStream bStrm = new ByteArrayOutputStream();

imageData = bStrm.toByteArray();

im =Image.createImage(imageData, 0, imageData.length); // Create image from the byte array

In this case there was only one class implemented for the Benchmark application. Figure 4.2 shows flow of the work in the symbian Benchmark application. The flow chart has some common components with the Android Benchmark flow chart shown above. The difference between them is on the conditions which guide the works done.

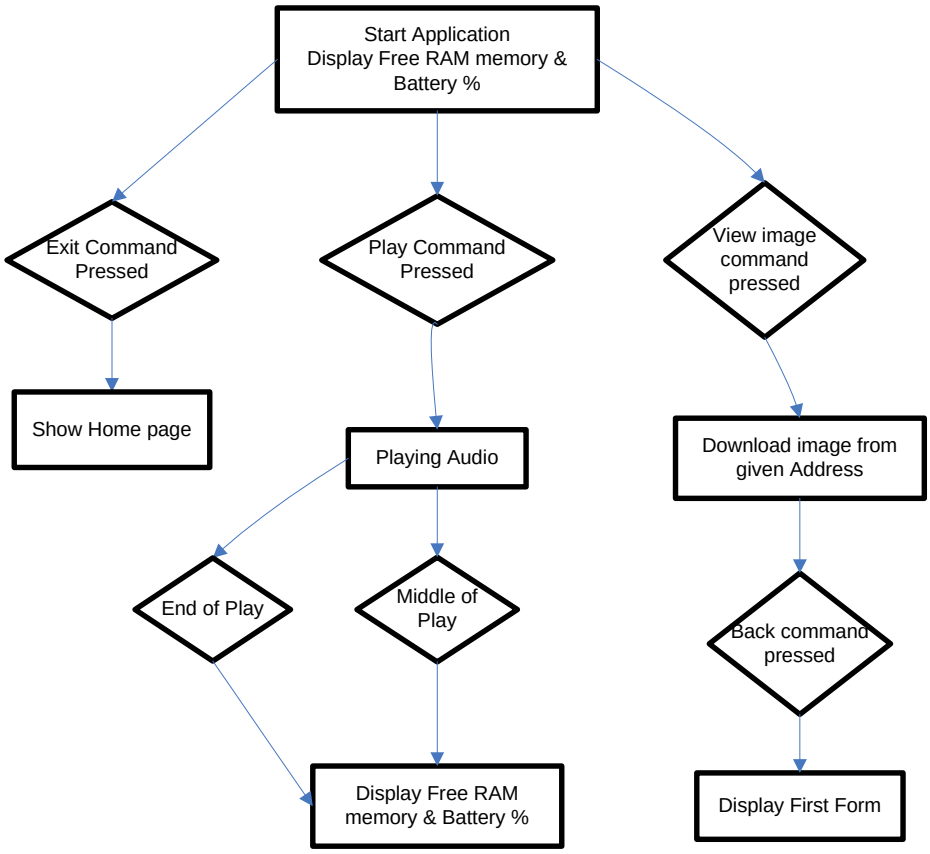


Figure 4.2: Symbian Benchmark Flow chart

Blackberry Benchmark application

The Benchmark application was developed using java micro edition. As previous applications Android and Symbian the work done here was similar. The only thing that differs was the way methods are called. To start with creating a project click on File menu-New-Blackberry project was done. Then Next button was clicked, project name was filled and the wizard was finished. The next step was creating class; being in the package New Class was created. The first work in the application was recording initial metrics information of Battery and memory status. In this application these works were done using the methods below:

BatteryPercent = DeviceInfo.getBatteryLevel(); [28]

memfree=Memory.getRAMStats().getFree();[29]

Then after, playing media continued and at the middle and end of playing media metrics status was recorded.

For the second work of downloading image file there was a method DownloadImage that was invoked in parallel with the other work (i.e. playing audio). The code fragment is shown below.

private Bitmap DownloadImage(String URL). This method also called OpenHttpConnection method for establishing internet connection. Its code fragment is shown below.

```
InputStream in = null;

in = OpenHttpConnection(URL);{

HttpConnection c = null;

c=(HttpConnection)Connector.open(urlString);}
```

As in the case of Android application the image is read as Bitmap data type. This is done inside the method DownloadImage

```
Bitmap bitmap = null;

bitmap =Bitmap.getBitmapResource(in.toString());
```

In the application there was a second class that is used for displaying information needed as output. The class was declared as shown below.

```
public class Displayscreen extends MainScreen implements PlayerListener
```

In the class there was a lablefield. Every information that is seen as output was first sent to the lablefiled for displaying. It is declared as below.

```
private LabelField lab;

add(lab); // Adding the labelfiled to the main screen.
```

Figure 4.3 shows components and flow of the work in Blackberry benchmark application.

Displaying required information of the metrics measured in the required place and downloading image file were basic works. Starting second work in parallel and listening audio file middle and end position were conditions which guides the work through.

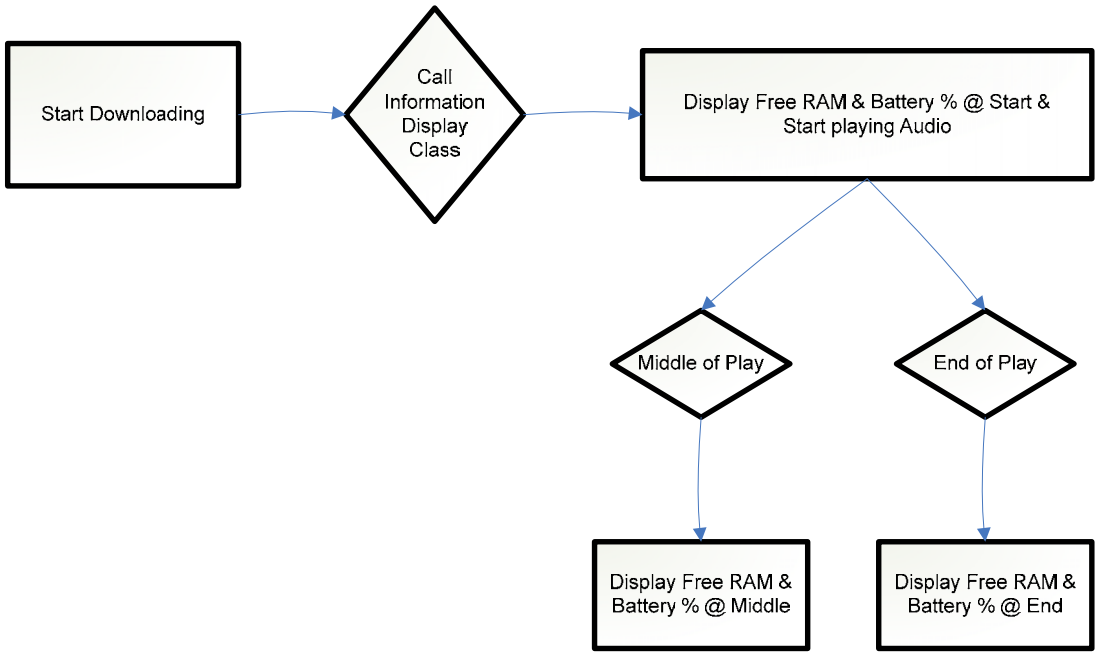


Figure 4.3: Blackberry Benchmark Flow chart

Windows Benchmark application

In Windows Benchmark application the software environment and the language used was different from the previous Benchmark applications. The environment for developing the application was Microsoft visual studio 2008 with windows mobile 6 SDK installed.

The language used to develop application was C#. To start with creating project visual studio program was opened and the procedures were followed like this, click File-New-Project then Visual C# for project type and then Smart Device was selected -Smart Device project. Finally the wizard was finished by clicking Ok button.

The next coming step was selecting Target platform and .NET compact Framework version. For the Target platform there was list of SDKs that are installed in the computer, the SDK needed was chosen. And for the .NET Framework appropriate version was selected. By Clicking on Ok button this step was finished.

At the end of this step a Form inside the Windows mobile emulator was seen in the form of c# design view. By double clicking in the Form and writing the code in this file was continued.

In this work there was one main class that holds all works. In side this class there is another sub class that was used to get battery percent, it was declared as shown in the code below.

public class SYSTEM_POWER_STATUS_EX [30]

and there was a struct defined in side the main class that is used to get the memory status, it was declared as shown here.

public struct MEMORYSTATUS [31]

For the different works needed to be done there were different buttons that were related to different events. The buttons were Browse-for searching audio file saved inside the device, play- to start playing audio, Middle- to get data at the middle of playing, and Stop- to get data at the end of playing. Download- to star downloading image (i.e. the second work done in parallel with the first one).

Figure 4.4 is somewhat different from the flow charts of Android, symbian and Blackberry seen above. The language used to develop the application for windows Mobile was C# and it was Java programming in previous applications. Here the conditions were invoked when a button was clicked. All conditions were ready from the homepage of the mobile as the application loads to the system. The basic works done were almost similar with the previous applications.

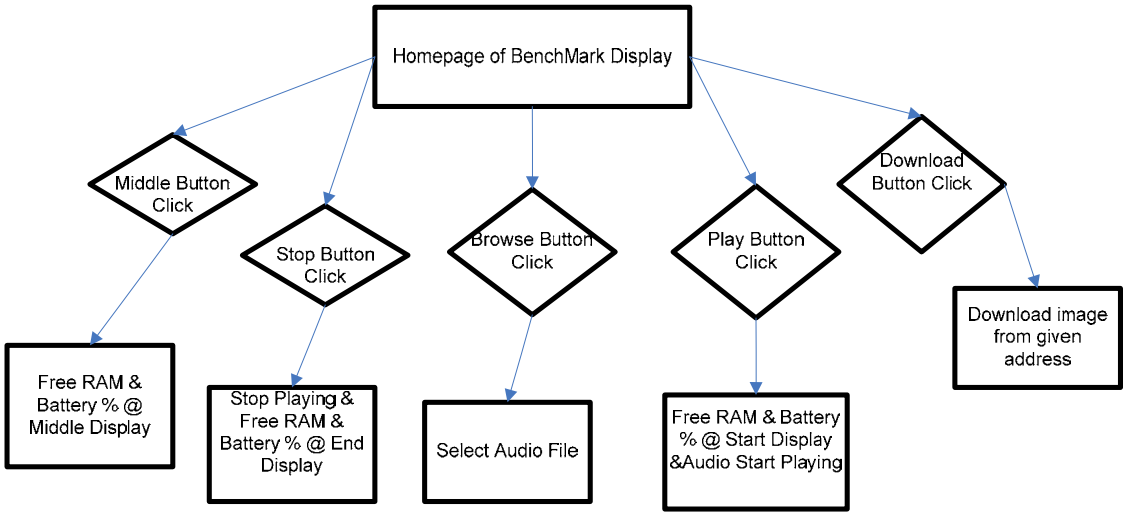


Figure 4.4: Windows Benchmark Flow chart

4.3 Metrics measured in the Benchmark

In the benchmark basic metrics that can be used for comparison of the performance of operating systems were measured. These metrics are: Memory management measured by free RAM status of the system at different points, Battery percent decreased by the work done, Time taken for completion of the work and ability of Multitasking.

Free RAM memory and Battery percent metrics were measured at three different points in the Benchmark; at the beginning of the work, in the middle of the work and finally at the end of the work.

The workload that is chosen in the Benchmark was playing long audio file and downloading image file from internet. The ability of multitasking of the operating system was tested if the works, i.e. playing audio and downloading image, in the Bench mark application can be done in parallel or not. Logically in taking free RAM memory at three different points the result should be less in the middle point and higher at the beginning and at the end. To get RAM memory taken by the work the difference between free RAM at initial and Middle points was calculated.

Similarly Battery percent should decrease from the beginning point to the middle and then to the end point. The total battery percent taken by the work was calculated by taking the difference of battery percent at initial and end points.

4.4 Implementation of the Benchmark

In this section implementation of the Benchmark application in emulator and real device is discussed. Each application has been implemented in their respective emulator. To run the applications the emulators need internet connection for downloading image file from the internet. Each emulator has its own technique to connect to internet. Android emulator need a batch file that is run before the application runs that connects emulator with the current connection type. For example if it is required to connect to proxy server with given name and port, simply write the following code in command prompt and run.

```
emulator -avd myavd -http-proxy proxy.aau.edu.et:8080
```

This code initialize emulator with the given proxy addresses and port name. This was done in this thesis work.

For symbian emulator connection was set inside the emulator by fixing host name and host port as needed.

Blackberry Emulator internet connection was enabled by setting host name and proxy in side the file rimpublic, this file is found inside Blackberry plug-in folder. The lines added in the file are:

application.handler.http.proxyEnabled=true
application.handler.http.proxyHost=proxy.aait.edu.et
application.handler.http.proxyPort=3128.

Then the application was run after the MDS emulator has been launched. For Windows emulator address of host name and port was set inside the emulator and additional software (i.e. Windows Mobile Device Centre) should be installed for the connection of internet.

Implementing the application in emulator did not give true data as in real devices. For example the memory capacity and battery percent read from the emulator were not expressing real data of metrics status.

Implementing on real device need the mobile device with appropriate operating system installed and the device should be connected to the computer as removable devices. For the connection some of them need software to be installed. Implementation on real device was a hard problem. To implement on real devices each application need its own device that is installed with the required operating system. The operating system chosen to be compared was the recent versions. In addition each device hardware performance should have nearly similar. RAM memory capacity and CPU speed were considered to be nearly equal in selecting actual devices. This specification helps to compare feature performance in nearly similar platforms. These limited specifications and the recent versions of the operating systems made the implementation of the benchmark very difficult.

Furthermore the price of devices were very expensive, this is because of their new operating systems version. One device costs 12,000 birr locally and this really needs much budget.

But finally executable files were installed on actual devices which have the required device specification. With the result found conclusion were drawn by comparing metrics values.

5. RESULTS AND DISCUSSION

In this chapter results found on emulator software and results from real device is discussed. Results of emulators were captured when the benchmark application run on the emulator and display results at different time of the work.

The audio file selected has 6.11MB size, its duration of play is 6 minutes and 40 second. Audio file playing was selected because all the SDKs installed support the feature. Any common workload can be chosen for the benchmark application.

Internet connection speed affects the time taken for downloading image file. When the connection was not established well or if there was error in downloading the file, error message was displayed in time limit. Different sizes of image and audio file have effect on the metric values. As size increased Power and memory taken to accomplish the work increased.

CPU speed and RAM size of the devices have also a major consequence on the output result. As CPU speed increases the time taken for the application decreases; this makes power and memory taken to be decreased.

This work clearly compared OS features. To do so the devices chosen should not have different hardware platform specification. In this case the effect of the workload on the OS was seen effectively.

The applications can be used to test different devices with different hardware specifications and OS. It can also be tested on devices with varied specification that support the same OS. For instance different types of devices that run Android OS 2.2 can be compared.

5.1 Android Results

The pictures followed are Android emulator results. Figure 5.1 a is the view of the emulator home page at start Figure 5.1 b is when metrics information is read initially and the smaller button was used to play audio file and Next button was used to start downloading image file. Figure 5.1 c adds middle point information of the metrics read from the emulator system. Figure 5.1 d adds the readings of metric values at the end of playing audio. And finally Figure 5.1 e is the image downloaded.

Figure 5.1: Android emulator results

5.2 Symbian Results

Figure 5.2 shows symbian emulator results one by one. Figure 5.2 a shows the emulator homepage at start. Figure 5.2 b shows the initialization of the emulator with its listening port. Figure 5.2 c & d shows SDK progress steps. If there is error in the application or in the software the SDK progress stops and shows error message. After the progress in SDK is done Figure 5.2e is shown. In this step access point for the emulator should be selected. Figure 5.2 g was shown when the application output start displaying. The data here was the readings taken at the initial point in the benchmark application. By choosing either play audio or image download command the works can be started. Choosing the first work of interest first then after, the second work has been chosen immediately after to test multitasking ability.

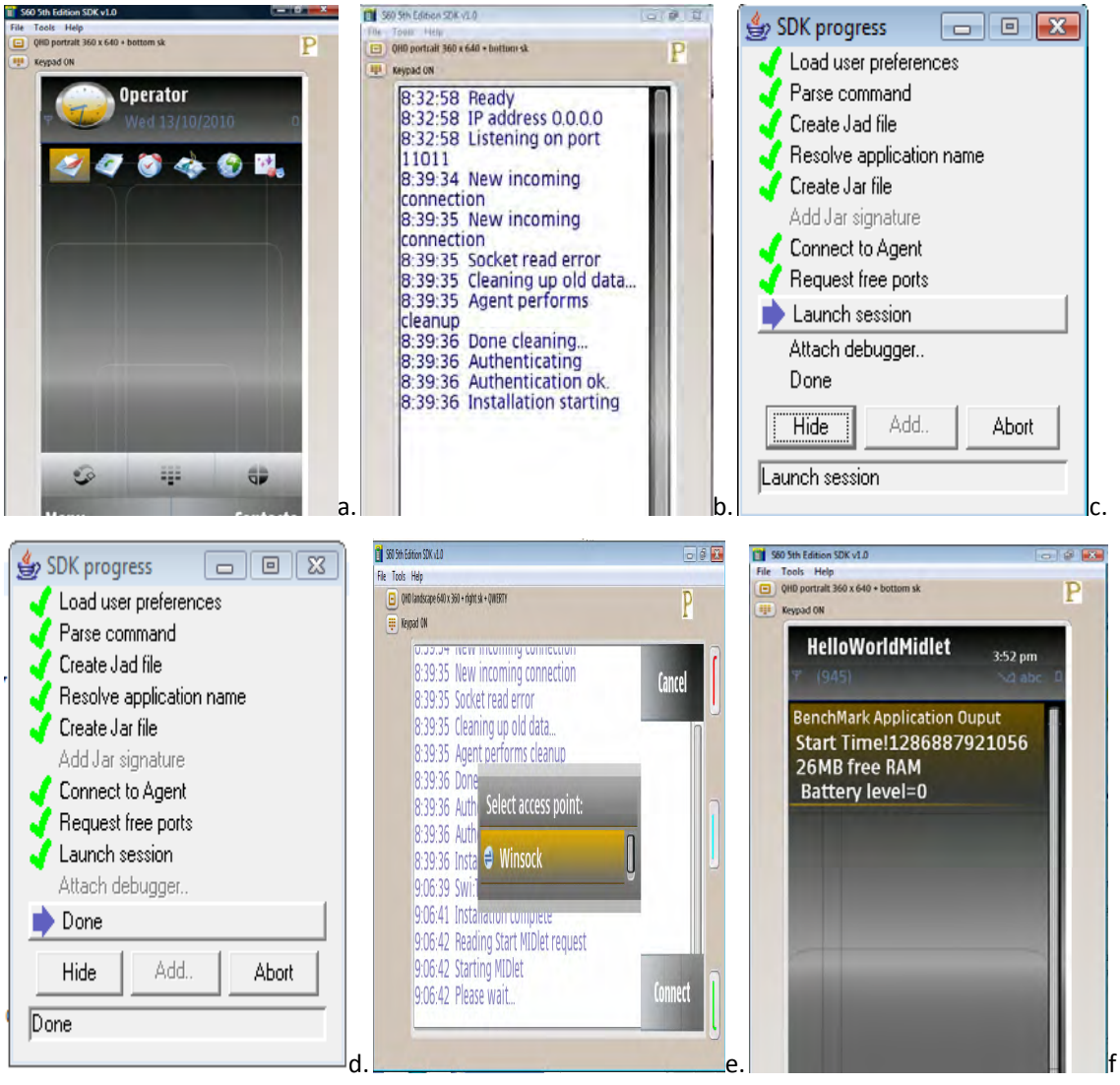
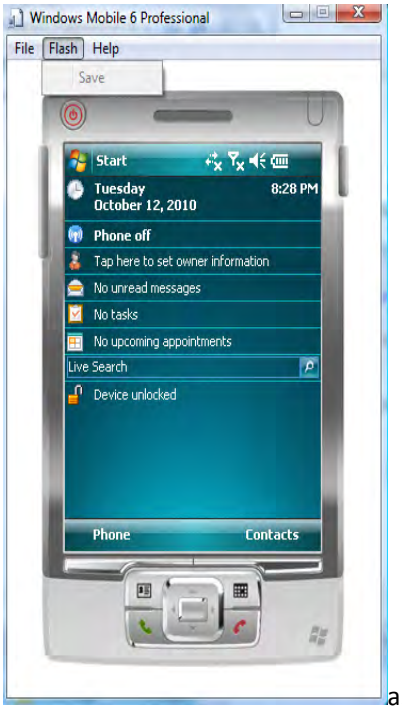


Figure 5.2: Symbian emulator results

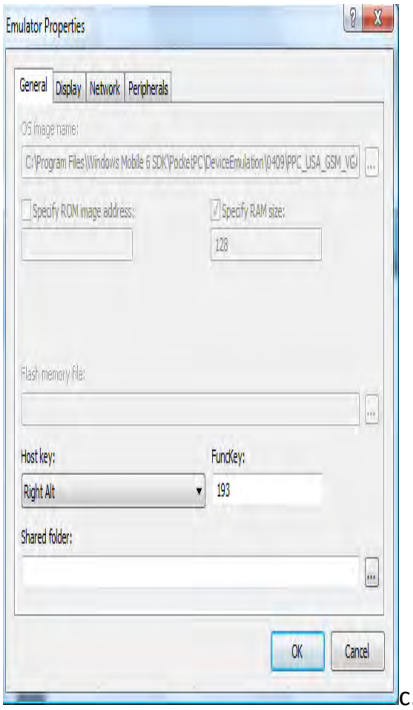
5.3 Windows mobile Results

Figure 5.3 shows windows mobile emulator results. Figure 5.3 a is the home page of the emulator at start. Figure 5.3 b is the application window with different control buttons and textbox. An audio file required was first configured with the device, this was done by clicking File menu and select configure and choose a folder with audio file from the required location. This is shown in Figure 5.3 c. Once the audio files were configured to select audio file from the device Browse button was clicked and the required file was chosen to play. This is shown as in Figure 5.3 d. Then when play button was clicked the application displayed Memory and Battery status found at the start and audio file started to play. Figure 5.3 e shows free Memory and battery percent at the beginning of the work. As the time is reached and Middle button is clicked the values of the metrics measured at this time is displayed. Figure 5.3 f shows middle information as Middle button is pressed. Finally Figure 5.3 g is shown when audio playing was finished and Stop button pressed.



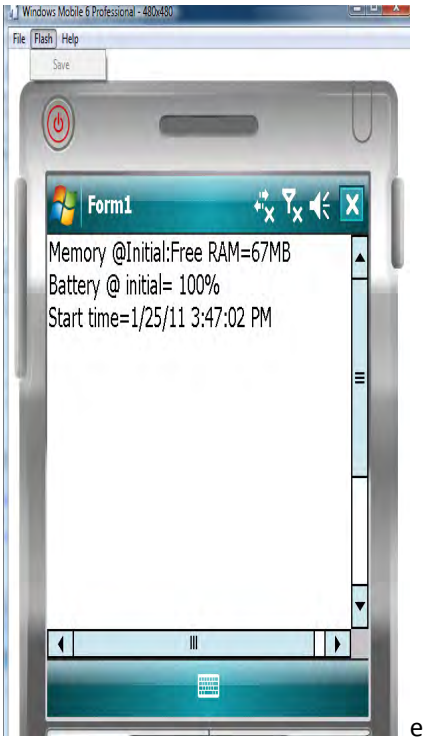
a.

b.



c.

d.



e.

f.

g.

Figure 5.3: windows Mobile emulator results

5.4 Blackberry Results

In the following figures Blackberry simulator results are shown step by step. Figure 5.4 a is the home page of the simulator and Figure 5.4 b was shown when the application finished running. To see output results the application name has been searched and clicked. In Figure 5.4 c readings of the metrics at initial, middle and final points are shown at a time in a page.

a. b. c.

Figure 5.4: Blackberry simulator results

6. PERFORMANCE ANALYSIS

6.1 Emulator and real device Results

In this chapter performance of each operating system feature is going to be evaluated based on the Benchmark application output. The Benchmark application measured RAM memory used, battery percent decreased, time taken for the application and test multitasking ability. Based on each feature performance of each operating system is analysed below.

Android results found from the emulator as shown in Figure 5.1 d gives the following values:

Memory@ initial: 35MB Battery @initial: 50%

Memory @Middle: 34MB Time Taken: 141 second

Memory @ End: 34MB Battery @ End: 50%

Data found from emulator didn't show real values of the metrics. But the logical fact seems the same with real devices; because as seen in the values above the free RAM memory decreases from initial to middle point.

The android application was tested on real devices. The first device found was Samsung SGH-T959 Galaxy S Vibrant with Android OS 2.2, RAM size of 512MB, and CPU speed of 1 GHZ. The data found are as shown below:

Memory@ initial: 92MB Battery @initial: 26%

Memory @Middle: 90MB Time Taken: 213 second

Memory @ End: 98MB Battery @ End: 22%

The second device that was used to test on is LG P500 Optimus one. It supports Android 2.2 OS. It has CPU speed of 600MHZ and RAM 512MB. The data found indicates the following value:

Memory@ initial: 84MB Battery @initial: 92%

Memory @Middle: 81MB Time Taken: 217 second

Memory @ End: 84MB Battery @ End: 89%

Symbian results found from emulator indicate the following values for the metrics:

Memory@ initial: 26MB Battery @initial: 0%

Memory @Middle: 19MB Time Taken: 153second

Memory @ End: 22MB Battery @ End: 0%

The Benchmark application is run on real Smartphone device. The device is Sony Ericsson Satio U1i/Idou(SE Satio), that supports symbian OS 9.4 series 60 5th edition (symbian ^1). It has CPU speed of 600 MHZ and RAM size of 256 MB. It returned the following data:

Memory@ initial: 76MB Battery @initial: 64%

Memory @Middle: 71MB Time Taken: 222 second

Memory @ End: 74 MB Battery @ End: 58%

Blackberry Benchmark application was run on the simulator. The output was shown in Figure 5.4c and it returned the following data:

Memory@ initial: 11MB Battery @initial: 100%

Memory @Middle: 16MB Time Taken: 122 second

Memory @ End: 16MB Battery @ End: 100%

The Benchmark application for Blackberry Smartphone was tested on real device. The device is RIM Blackberry Bold 9700(RIM Onyx) with Blackberry OS 5.0. It has 624 MHZ CPU speed and 256 MB RAM. The result found shows the following data:

Memory@ initial: 80MB Battery @initial: 44%

Memory @Middle: 74MB Time Taken: 236 second

Memory @ End: 78MB Battery @ End: 35%

The Benchmark application for windows Mobile OS was run on emulator. The results found as shown in Figure 5.3g indicates the following values:

Memory@ initial: 67MB Battery @initial: 100%

Memory @Middle: 48MB Time Taken: 115 second

Memory @ End: 48MB Battery @ End: 100%

6 Performance Analysis

The windows Mobile OS application was also run on real Smartphone device. The device chosen was Psion Teklogix Ikon 7505 WWAN that supports windows mobile 6 professional operating system and CPU speed of 624 MHZ and 256 MBRAM size. The application output has shown the following results.

Memory@ initial: 87MB Battery @initial: 87%

Memory @Middle: 79MB Time Taken: 230 second

Memory @ End: 87MB Battery @ End: 80%

Devices taken in to consideration are LG P500 Optimus one, Sony Ericsson Satio U1i/Idou (SE Satio), RIM Blackberry Bold 9700(RIM Onyx) and Psion Teklogix Ikon 7505 WWAN.

As much as possible these devices are chosen based on their CPU speed and RAM memory. As shown in the table below each feature are nearly similar for each device.

Table 6.1: Device specifications

	Supported OS	CPU speed (MHZ)	RAM memory (MB)
LG P500 Optimus one	Android 2.2	600	512
Sony Ericsson Satio U1i/Idou	Symbian^1	600	256
RIM Blackberry Bold 9700	Blackberry OS 5.0	624	256
Teklogix Ikon 7505 WWAN	WM 6 professional	624	256

6.2 Memory taken by the application

Table 6.2: Memory taken on emulators

	Memory Initially(MB)	Memory at middle of work(MB)	Memory at end of work(MB)	Difference b/n Initial & Middle (MB)
Android	35	34	34	1
Symbian	26	19	22	7
Blackberry	11	16	16	5
Windows	67	48	48	19

In the previous chapter memory metric measured from emulators indicates free RAM memory at three different points; initial, middle and end of the work. For instance from Table 6.2 the Android emulator shows free RAM memory 35MB, 34MB, and 34MB for initial, middle, and end points respectively. The memory taken in between the start and middle points is 1MB.

The end point status also indicates the same memory level as in the middle point. This value indicates the memory taken for the work did not released yet. In same manner the other emulators also shows the same logic as Android emulators.

As shown in Table 6.2 each emulator shows different memory taken by the application. Except BlackBerry emulator initial free memory is higher, and after the work started Memory at middle of work shows some decrement and finally when the work is completed memory might be released and free memory is increased or remains the same as middle point value. In comparison of memory taken by the work, difference between initial free RAM memory and middle point free RAM memory was considered. End point free RAM memory shows the free RAM memory after the work is completed. Based on the result from emulator the list of OSs are Android, Blackberry, Symbian and Windows Mobile in order from less to high memory taken in between initial and middle points of the work.

Table 6.3: Memory taken on real devices

	Memory Initially(MB)	Memory at middle of work(MB)	Memory at end of work(MB)	Difference b/n Initial & Middle (MB)
Android	84	81	84	3
Symbian	76	71	74	5
Blackberry	80	74	78	6
Windows	87	79	87	8

Memory Used by the application when seen on real devices is shown above. Based on this result the OSs Android, Symbian, Blackberry, and Windows Mobile are listed in order of memory taken from less to high. The reasons for each data behind are discussed below. For the case of Android OS it uses special virtual machines to run each application, dalvik VM. Dalvik VM is designed to be very memory efficient, being register based (instead of being stack based like most Java VMs) and using its own byte code implementation. The Dalvik VM makes full use of Linux for memory management and multi-threading, which is intrinsic in the Java language. Symbian has implemented the demand paging memory management technique to reduce the time required to retrieve data. As the RAM in a mobile phone device is considerably smaller than that of a conventional computing device, this method works very well. Demand paging has been implemented in certain versions of the operating system, namely versions 9.3, 9.4 and 9.5. Demand paging works through page faults; a page fault occurs when the CPU requests a page, and it is not in one of the active memory slots. The requested will then be fetched from the storage, and will then continue to reside in one of the active slots. As this process continues, through various page faults, all the slots will eventually get filled. This algorithm works extremely well if these are the pages required the most frequently. The rate of page faults decrease as the pages are found with the active memory itself, reducing retrieval time considerably. If a page fault occurs, one of the pages is removed and the new one is inserted in its place. [32]

In windows Mobile each process has to be fitted into the 32MB slot, that is slot 0 reserved for the currently executing process. If the process needs or tries to allocate more memory, the memory allocation fails. In this OS there is a limited memory management. [33]

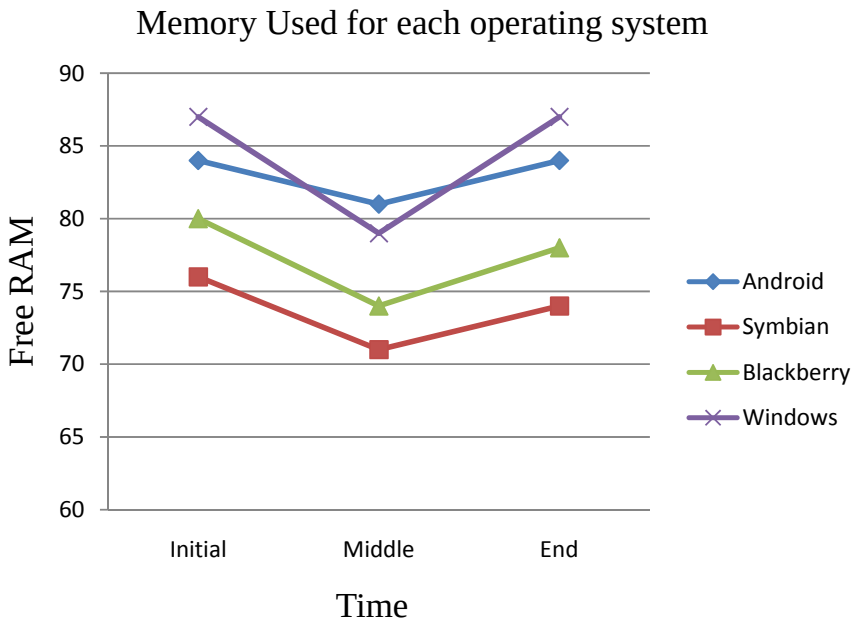


Figure 6.1: Plot of Free RAM versus Time

The Figure 6.1 shows previous data in graph format. The windows Mobile system shows a great decrement and Android System shows less decrement of Free RAM memory from initial to middle time. Android memory management shows best performance in handling workloads.

6.3 Time taken by the application

Table 6.4: Time taken by the application on emulators and real devices

	Time taken on Emulator(second)	Time taken on real device(second)
Android	141	217
Symbian	153	222
Blackberry	122	236
Windows	115	230

Data in Table 6.4 shows time taken by the application on each type of devices. Operating systems based on emulator result are listed as Windows Mobile, Blackberry, Android and Symbian in order from less to high time taken. Based on the data found from real devices their order is Android, Symbian, Windows Mobile and Blackberry.

The work given was playing audio file in parallel with downloading image file, for this work the longer part was playing audio file. Downloading image was done in shorter time than playing audio. The factors behind the results are the speed of the machines, capability of handling works and capability of playing audio file. For the case of Windows Mobile and Blackberry capability of playing audio determined the time taken. Windows mobile is the champ of multimedia capabilities than Blackberry. So time taken in windows Mobile is less than Blackberry. But Android and symbian can handle workloads easier. Especially android performance with respect to time taken for the application is best.

6.4 Battery percent decreased in running application

Table 6.5: Battery percent on real devices

	Battery percent at initial	Battery percent at End	Difference b/n Initial and End
Android	92	89	3
Symbian	64	58	6
Blackberry	44	35	9
Windows	87	80	7

The battery percent shown in the emulators were not giving any relevant data, because it always returns the same value for every point of access. So for this feature the data on real devices was used. As shown in the Table 6.5 battery percent taken to complete the work is computed from the difference between batteries at initial to end. Based on the table Android, Symbian, Windows Mobile and Blackberry is in the order from less to high battery taken systems for the application. This order is similar with time taken values. As the time for doing the application is shorter the energy consumed is also less. Additionally each system supports different power management policy. Symbian OS favours a distributed approach to power management, with components at different layers of the OS responsible both for managing their requirements on system power. For Android system Dalvik VM adds a few distinct features that uniquely define the security and power-preserving model.

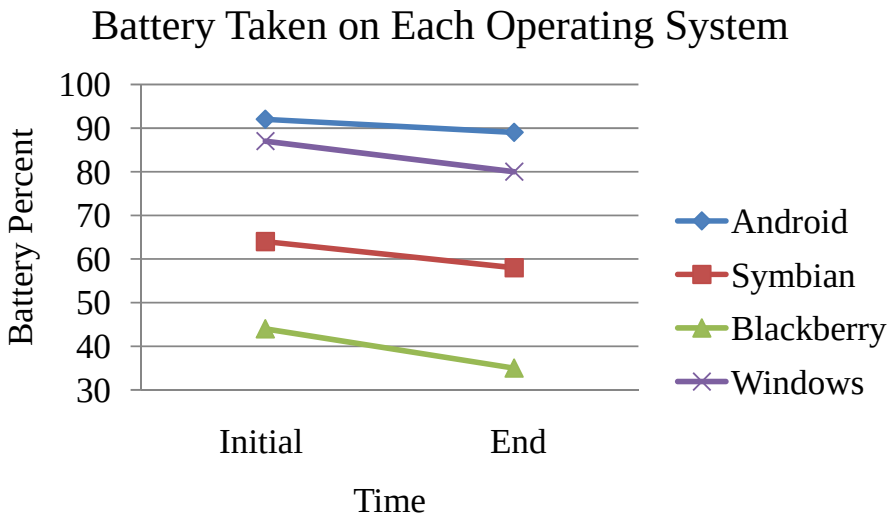


Figure 6.2: Plot of Battery Percent versus Time

Blackberry systems shows great change in battery percent and in contrast android systems shows less change of battery percent. Power management feature of Blackberry shows less efficiency according to the result. Android power management is best for preserving power of the system.

6.5 Multitasking Ability

This feature is the ability of doing more than one work simultaneously. In this regard all except windows Mobile 6 were effectively doing the given works at the same time. Google allows third party apps to use the multitasking feature of Android OS. If android users switch to another app, the OS looks after the previous app and makes it completely invisible to the user. Symbian OS implements pre-emptive multitasking. In this work window Mobile SDK v 6 was chosen to test application. Since multitasking is not supported in versions before windows 7.0 WM 6 cannot do simultaneous works. The Blackberry Smartphone application framework allows applications to remain running in the background (in sleep mode). So in regard to this feature only WM6 could not support multitasking.

6.6 Previous work on Benchmark

The LINPACK Benchmarks are a measure of a system's floating point computing power. Introduced by Jack Dongarra, they measure how fast a computer solves a dense N by N system of linear equations Ax = b, which is a common task in engineering. The solution is obtained by Gaussian elimination with partial pivoting, with $2/3 \cdot N^3 + 2 \cdot N^2$ floating point operations. The result is reported in millions of floating point operations per second (MFLOPS).

For large-scale distributed-memory systems High Performance Linpack is used as a performance measure for ranking supercomputers in the TOP500 list of the world's fastest computers. There is now also a Green500 list ranking the machines on the TOP500 list based on energy efficiency, in FLOPs per Watt. The High Performance Linpack benchmark is run for different matrix sizes N searching for the size N_{max} for which the maximal performance R_{max} is obtained. The benchmark also reports the problem size $N_{1/2}$ where half of the performance ($R_{max/2}$) is achieved. [34]

The LINPACK Benchmark is customized for different mobile operating systems. There is LINPACK benchmark for Android, and Iphone systems.

7. CONCLUSION AND RECOMMENDATION

7.1 Conclusion

Finally in conclusion for this thesis work different major points can be raised. First of all it provided a brief understanding of special features of each operating system. The special features help the systems to behave in a different way than the other. The Benchmark application designed was specific for each system. The metrics obtained were used to evaluate performance of each feature. Memory management, power management, speed and multitasking ability of the system are fundamental features of the operating system. The benchmark tried to measure these features using appropriate metrics. The metrics are RAM memory taken by the workload given, battery percent decreased because of the workload done and time of completion of the work.

Based on each feature better performance is seen in Android operating system. The major reasons for this result are being Linux based system and the Dalvik VM. These two points help android system to perform best in features considered. The next better system seen is Symbian operating system. Its distributed power management and demand paging memory accessing policy helps it to perform better in features seen. The third system that is performing well in features is Blackberry operating system. Windows Mobile is at the last of the list in performance regard. Blackberry and windows mobile are closed system, which means internal structures of the system are not revealed. With this reason detail policy for memory management and power management is not known.

The second major advantage of this work is to identify the best systems and the reasons behind. Hence those systems that are lower in the list can be updated by considering best feature policies in better performed systems.

Third profit with this work was to be familiar with different environments for developing mobile applications. Different developing techniques were used for each Benchmark application.

Different limitations were faced during the work. Some of them are listed below.

- Getting the right SDK and the required software for the SDK to work properly.

  Being familiar with each development environment

  - Variety of developing language for different environment
  - Mobile SDKs made the computer so slow and Stuck now and then ,this was very uncomfortable
  - Results found from emulators are not real data to draw conclusions. So searching actual device to get real results was the hard challenge.

7.2 Recommendation

In this section future works that can be extended from this thesis work are described. Nowadays developing mobile applications that are used for different purpose is becoming ordinary. Based on this thesis work one who is interested in the area can continue its work in the following different titles.

- Propose feature update policy for Operating systems.

In this title one can select features that are not good from operating systems. Then try to make them to be more reliable and efficient for different workloads. Those performing less efficient in their feature can be chosen and feature policy of better systems can be customized for them.

- Develop benchmark applications that can measure performance of the system in a way different from this work.

With this work it is expected to measure performance of operating systems. The metrics going to be measured can be any features of the system. The work load is going to be chosen to challenge the selected metrics. Then the benchmark application should give a reasonable measured output that can check performance of systems.

- Adding new feature for Open Operating system

Android and Symbian are Open Operating systems. This helps developers to add new features for the system.

REFERENCES

[1] Daniel McKenna, “Mobile Platform Benchmarks: A Methodology for Evaluating Mobile Computing Devices”, Transmeta Corporation, January 2000

[2] Wikipedia, the free encyclopedia- Mobile Operating System, http://en.wikipedia.org/wiki/ MobileOperating System (accessed September 2010).

[3] “Mobile OS shootout: iPhone, Blackberry Storm, Palm Pre, HTC, Symbian s60”, March 21 2009

[4] Rashad Maqbool Jillani, “Mobile OS Security”

[5] Benjamin Speckmann. “The Android mobile platform” - A Review Paper Submitted to the Eastern Michigan University Department of Computer Science In Partial Fulfillment of the Requirements for the Master of Science in Computer Science, Approved at Ypsilanti, Michigan, April 16, 2008

[6] Jane Sales et al, “Symbian OS Internals”-Real-time Kernel Programming, Published by John Wiley & Sons, Ltd. 2005

[7] Gregory F. Welch, “A Survey of Power Management Techniques in Mobile Computing Operating Systems”, Department of Computer Science University of North Carolina. Chapel Hill, 1995

[8] Marc A. Viredaz et al, “Energy Management on Handheld Devices”. HP Laboratories Palo Alto. August 28th, 2003

[9] Colin Walls, Embedded Software Technologist, “Embedded Systems White Paper”, Mentor Graphics Corporation, September 2009.

[10] Frank Maker and Yu-Hsuan Chan, “A Survey on Android vs. Linux”- Department of Electrical and Computer Engineering, University of California, Department of Computer Science.

[11] “Smartphones and Symbian OS”, (Accessed June 2010) http://media.wiley.com/product_data/excerpt/53/04700184/0470018453.pdf

[12] NOKIA GROUP, “Nokia and Symbian OS”, Nokia Mobile Phones. Finland,

[13] GradDip Students: Liam Barrett & Mary O’Riordan, “Blackberry Operating System”

[14] Grace Aquino, “Blackberry vs. Windows Mobile 6”, July 5, 2007 http://www.laptopmag.com/review/software/blackbbery-vs-windows-mobile-6.aspx

[15] Wikipedia, the free encyclopedia-Windows Mobile. http://en.wikipedia.org/wiki/Windows_Mobile. (Accessed August 2010).

Reference

[16] Bonnie Cha, “Review of Windows Mobile 6”, February 23, 2007.
http://www.zdnet.com.au/windows-mobile-6-339273811.htm

[17] Joel Ivory Johnson, Software Engineer. “Windows Mobile Power Management”, RDA Corporation. 2008

[18] “A pocket PC central Brief: Windows Mobile 6”, Copyright 2000-2007
http://pocketpccentral.net/wm6brief.htm

[19] Android developers, “Installing the SDK”,
http://developer.android.com/sdk/installing.html, (accessed July 2010)

[20] Forum Nokia, “S60 5th Edition SDK, Installation Guide Version 1.0”, January 2nd, 2009.

[21] Blackberry Forum, “Setting up Necessary Tools for Blackberry”, (accessed September 2010)
http://supportforums.blackberry.com/t5/Java-Development/Setting-up-Necessary-Tools/ta-p/442687

[22] Microsoft Download Center, “Windows Mobile 6 Professional and Standard Software Development Kits Refresh”, (accessed August 2010)
http://www.microsoft.com/downloads/en/details.aspx?FamilyID=06111a3a-a651-4745-88ef-3d48091a390b.

[23] Stack overflow-asked by Maxood, “How to obtain Battery Stats in Android at runtime?” February 15, 2010.
http://stackoverflow.com/questions/2266065/how-to-obtain-battery-stats-in-android-at-runtime

[24] Android developers, “Public class ActivityManager class overview”.
http://developer.android.com/reference/android/app/ActivityManager.html (Accessed June 2010)

[25] Wei-Meng Lee, “Connecting to the Web: I/O Programming in Android”
.http://www.devx.com/wireless/Article/39810/1954. (Accessed July 2010)

[26] Forum Nokia, “Nokia-specific system properties”
http://library.forum.nokia.com/index.jsp?topic=/Java_Developers_Library/GUID-A1F90B63-5F14-4829-BCA1-A23B925CF670.html . (accessed August 2010)

[27] Andreas Jakl, “Java™Platform, MicroEditionPart 7GenericConnection Framework”. v3.0 –25 ,April 2009

[28] RIM Device Java Library. “Class DeviceInfo”.
http://www.blackberry.com/developers/docs/4.0.2api/net/rim/device/api/system/DeviceInfo.html.
(accessed July 2010).

Reference

[29] RIM Device Java Library. “Class Memory”. http://www.blackberry.com/developers/docs/4.0.2api/net/rim/device/api/system/Memory.html (accessed July 2010).

[30] Geoff Schwab and Jonathan Wells. “How to Get the Device Power Status”. Microsoft Corporation.October, 2003. http://msdn.microsoft.com/en-us/library/aa457088.aspx

[31] MSDN Library, “MEMORYSTATUS structure”, http://msdn.microsoft.com/en-us/library/aa366772(v=vs.85).aspx. (Accessed September 2010)

[32] Karishma Sundaram, “A Guide to Demand Paging and How it Improves Memory Management on Mobile Phones”, Jan 21, 2010

[33] Windows Live, “Memory Architecture of Windows Mobile 5.0.”, December 24 2009

[34] Wikipedia, the free encyclopedia- LINPACK. http://en.wikipedia.org/wiki/LINPACK. (Accessed October 2010).