



UNIVERSITY OF NAIROBI

SCHOOL OF COMPUTING AND INFORMATICS

Middleware for Mobile Phone Grid based on Android

By

Mugambi Jeremiah Ananga

P58/73202/2009

Supervisor

Mr. E. Miriti

October 2011

University of NAIROBI Library




0378952 6

Submitted in partial fulfillment of the requirements of the Master of Science in
Computer Science

Declaration

This project as presented in this report, is my original work and has not been presented for any other University Award.

Signed: 

Date: 18/10/2011

Mugambi Jeremiah Ananga.

P58/73202/2009

The project has been submitted as part fulfillment of requirements for the Masters of Science in Computer Science of the University of Nairobi with my approval as the University supervisor.

Signed: 

Date: 19/11/2011

Mr. E. Miriti

Project Supervisor

School of Computing and Informatics

University of Nairobi

Abstract

The mobile phone has been dubbed the “the single most transformative tool” for development, its usage has increased coupled with lots of technological advancement. This dissertation describes and reports on mobile grid middleware implementation. The advances in mobile devices promise ubiquitous computing. Smart mobile phones have relative high computing power which is not fully utilized since they are only used for making call and text messages.

In this project we develop a middleware framework for grid computing based on android operating mobile system, on which applications could be built. This middleware acts as a proof of concept that grid computing can be implemented on mobile phones. The grid middleware runs on android mobile platform, addressing mobility and resource sharing in mobile grid environment harnessing the raw power of mobile devices which are not fully utilized.

The methodology used deployed prototyping where agile methodology was used resulting to “inspect-and-adapt” approach to development that greatly reduced development time and delivered working modules.

The middleware comprised of the server and client node applications. The communication was based on WIFI and successfully tested over mobile telephony network. The middleware communication and task handling was achieved. The task handling involved the following; ability of client to submit a task request, to the coordinator, the coordinator generate computational load, the server ability to split the task and to submit subtask work to client for computation and results returned back and finally the coordinator aggregating the results and sending to the requesting client.

There was successful testing of the middleware for both nodes that is server and client on emulator and finally using two Ideos android phones in real work environment. The communication was successful over mobile operator network where WIFI was not available, proving viability of grid middleware and acting as a proof of concept that grid computing is possible on mobile phones.

Acknowledgement

I would like to thank my supervisor, Mr. E Miriti for his continuous guidance and enormous support during my project period. Thanks to the University of Nairobi, School of Computing and Informatics' Management and Academic staff for all support provided. I am indebted to Ms. Masinde as she initially proposed this project.

I appreciate the support from my dear wife Kellen and my lovely son Alvin who inspired and prayed for me from the start to the end of the course. Thanks to my caring parents and all my siblings who encouraged me.

To the fellow Msc. Computer Science students who were involved in testing the system and to everyone who contributed to the success of this project, I say thank you and God bless you all.

TABLE OF CONTENTS

Declaration	i
Abstract	ii
Acknowledgement.....	iii
List of Acronyms.....	x
CHAPTER 1: INTRODUCTION.....	1
1.0 Background	1
1.1 Definitions of Important Terms.....	1
1.2 Problem Statement.	2
1.3 Problem Justification.....	2
1.4 Research Contribution.....	3
1.5 Objectives.....	4
1.5.1 Overall objectives.....	4
1.5.2 The sub objectives	4
1.6 Research Question.....	4
1.7 Assumptions and limitations of the research	4
1.8 Organization/ Structure of the Report	5
CHAPTER 2: LITERATURE REVIEW	6
2.0 Grid Computing Overview.....	6
2.1 Characterization of Distributed Systems.....	6
2.2 Mobile Grid Computing.....	8
2.3 Previous work done on Mobile Middleware	8
2.4 Android Mobile OS.....	10

2.5	Why Android?	12
2.6	Conclusion.....	13
CHAPTER 3: METHODOLOGY		14
3.0	Chapter Overview	14
3.1	Methodology Overview.....	14
3.1.1	Agile Methodology.....	14
3.1.2	Limitation of the Methodology.....	14
3.2	Requirement Specification and Analysis	15
3.3	Middleware Functional Requirements	15
3.4	Non Functional requirements	16
3.5	Hardware and Software Requirements.....	17
3.6	Communication Framework.....	17
3.6.1	Socket Communication.....	17
3.6.2	Peer to Peer communication	19
3.7	Operation of the middleware	19
3.8	Conclusion.....	20
CHAPTER 4: SYSTEM DESIGN PHASE.....		21
4.0	Chapter Overview	21
4.1	Middleware Architecture Overview.....	21
4.1.1	Proposed System Detailed Design.....	21
4.2	Node Functions	22
4.3	Middleware integrated with android software stack	22
4.4	Android Applications	23
4.5	Modules Design.....	24

4.6	Test Case Scenario	24
4.6.1	Test Case process:	25
4.7	Grid Operation Illustration.....	25
4.8	System High Level Diagrams	27
4.8.1	Activity Sequence Design Diagram.....	27
4.8.2	Android Activity Diagram.....	27
4.9	Conclusion.....	29
CHAPTER 5: IMPLEMENTATION		30
5.0	Chapter Overview	30
5.1	Implementation Environment.....	30
5.2	Project Coding.....	31
5.3	Permissions.....	31
5.4	Module Development.....	32
5.4.1	Main Graphical User Interface	32
5.4.2	System Resources Module.....	32
5.4.3	Server Module:	32
5.4.4	Client Module:.....	33
5.4.5	Interface Module:	33
5.5	Conclusion.....	33
CHAPTER 6: TESTING AND RESULTS.....		34
6.0	Chapter Overview	34
6.1	Testing.....	34
6.1.1	Unit Testing.....	34
6.1.2	Integration Testing.....	34
6.1.3	Performance Testing.....	34

6.1.4	Testing process	35
6.1.5	Emulator Setup and Testing	35
6.1.6	Testing and Evaluation Matrix	36
6.2	Results	38
6.3	Screen shots of the results and description.....	38
6.3.1	System Resources Results	38
6.3.2	Server Node Operation	39
6.3.3	Client node Operation.....	40
6.4	Conclusion.....	42
CHAPTER 7: DISCUSSION		43
7.0	Chapter Overview	43
7.1	Achievements	43
7.2	Challenges	44
7.3	Limitations of the System.	44
7.4	Recommendations for Further Work.....	45
7.5	Conclusion.....	46
REFERENCES.....		47
APPENDICES		49
Appendix A: User Guide.....		49
Appendix C: Sample Source Code.....		53

List of Figures

Figure 1: ICT adoption in Kenya in Percentage	3
Figure 2: Example of a distributed system (Licia et al 2001 p4).....	7
Figure 3: Characterizations of mobile distributed systems.....	7
Figure 4: Android Architecture	11
Figure 5: Nodes TCP communication time diagram	18
Figure 6: Packet Data Units (PDU) Packaging Layer	19
Figure 7: Proposed middleware architecture	21
Figure 8: Detailed design of the middleware System.....	22
Figure 9: The diagram Android Software Stack.....	23
Figure 10: Modules of the middleware.....	24
Figure 11: Task Generation illustration diagram.....	26
Figure 12 : Middleware Sequence Diagram	27
Figure 13: Activity Lifecycle Diagram	28
Figure 14: Association diagram.....	29
Figure 15: Main Graphical User interface of the server and client.....	32
Figure 16: Module testing diagram on emulator then on phone.....	35
Figure 17: Emulator telnet and port redirection setup	36
Figure 18: Server and client emulator displaying results	36
Figure 19: System Resource diagram.....	39
Figure 20: Server node operation diagrams.....	40
Figure 21: Server node operation diagrams.....	41

List of Tables

Table 1: Android market growth	13
Table 2: Use case for Do work	16
Table 3: Testing summary	37

List of Acronyms

3G	: 3rd Generation Mobile Telecommunications
API	: Application Programming Interface
APK	: Android Package (APK) file
ARM V7	: Architecture Reference Manual Version Seven
AVD	: Android Virtual Device
CPU	: Central Processing Unit
GUI	: Graphical User Interface
ICT	: Information Communication Technology
IDE	: Integrated Development Environment
KNBS	: Kenya National Bureau of Statistics
LPC	: Local Procedure Calls
MOM	: Message Oriented Middleware
OGSA	: Open Grid System Architecture
OGSI	: Open Grid Service Infrastructure
OS	: Operating System
PC	: Personal Computer
RPC	: Remote Procedure Call
SDE	: Service Data Elements
SDK	: Software Development Kit
SMS	: Short Message Service
SOAP	: Simple Object Access Protocol
SU	: Super User
TCP/IP	: Transmission Control Protocol /,Internet Protocol

UML : Unified Modelling Language
WEP : Wired Equivalent Privacy
WIFI : Wireless Fidelity
WPA : Wi-Fi Protected Access
XML : Extensible Mark-up Language

CHAPTER 1: INTRODUCTION

1.0 Background

The use of mobile phone technology has revolutionised communication in developing countries. This is due to the availability and affordability of mobile phones coupled with characteristics like mobility and advances in mobile technology like computing power. As the use of mobile devices is increasingly predominant, utilization of technologies like grid computing, (Katarina et al, 2010) is inevitable thus need for a reliable grid middleware. These devices especially the smart phones and Personal Digital Assistants (PDA) have relatively high computing power hence presenting favourable arena for mobile grid computing. However, full utilisation of these devices is never achieved since they are often idle, experience temporary and unannounced loss of connectivity due to mobility (Capra et al 2002). These devices have inherent characteristics, battery life is finite, low bandwidth, mobility and storage space is constrained. These restrictions slow application execution, and hinder operability.

Despite the constraints, mobile devices resources can be extended in developing countries, to develop mobile phone-based grids. These grids can be utilised to run various computing resource intensive applications like m-commerce, e-health, e-learning and e-environment monitoring.

Grid computing can benefit from clustering mobile devices forming a mobile grid by use of middleware. The middleware technologies provide abstraction platform built on top of network operating systems, to enhance the design and implementation of distributed applications. In this research project, the key intention was to develop grid middleware to run on android mobile platform to addressing mobility and resource sharing in mobile grid environment (What's Android on android developer, 2011).

1.1 Definitions of Important Terms

Grid Computing: Refers to combination of computer resources from multiple administrative domains to reach a common goal, it involves the aggregation of network connected computers to form a large-scale, distributed system for coordinated problem solving and resource sharing. Grid computing harnesses unused processing cycles, memory and other computing resources of all computers in a network for solving problems too intensive for any stand-alone machine, the resources are available to authorized users (Bart et al 2005).

Mobile Computing: Refers to a form of computing where portable devices are used, it involves being able to use a computing device even when being mobile and therefore changing location. Portability is key aspect of mobile computing

Mobility: Ability to change location. In this context computing device and user mobility is used in the project

WIFI: It is an acronym for "Wireless Fidelity." Refers to wireless networking technology that allows computers and other devices to communicate over a wireless signal.

Socket: Refers to one endpoint of a two-way communication link between two programs running on the network. A socket is bound to a port number so that the TCP layer can identify the application that data is destined to be sent

API: Refers to Application Program Interface, which is a set of program modules and tools for building software applications.

1.2 Problem Statement.

The increasing popularity of portable computing devices and advances in wireless networking technologies are facilitating the construction of new classes of distributed and mobile applications. However these mobile devices have not been utilised in a grid environment, which has been a reserve for traditional fixed grid network. Despite the challenges of mobility, reliance on battery power and heterogeneity of platforms there is high processing capability on most mobile devices that lay idle without being fully utilised. Most users tend to use mobile phones for making calls and sending messages only, while these phones are fully powered every day, this leads to the question;

How can mobile phones be used in a grid environment to harness the processing capabilities that many at times lie idle to run various applications by achieving mobility and resource sharing?

1.3 Problem Justification

The mobile phone has been dubbed the “the single most transformative tool” for development (Britni 2010). Kenya is among the rapidly growing mobile markets in Africa; the 2009 population census (KNBS 2010) highlight shows high adoption of ICTs with 63.2% number of households owning at least a mobile phone and 3.6% number of households owning at least one computer. At the end of 2007, there were 280.7 million mobile phone subscribers in Africa, representing a penetration rate of 30.4% (Wireless Federation 2011). This statistics leads to a need for a mobile middleware, to take advantage of the unutilized resources enhancing sharing and mobility in grid environment.

Figure 1 summarizes the statistics of ICT adoption for both mobile and computers in various from (KNBS 2010), which is the core motivation of this project. The high percentage of mobile penetration motivates the need to study problem of how best to utilise the mobile computing power.



Figure 1: ICT adoption in Kenya in Percentage

Source: KNBS. (2010). Census Highlight Brochure.

1.4 Research Contribution

The initial intended users are technologically savvy persons among the huge population who own mobile phones and researchers on the field of mobile grid computing. These users would actively utilise the grid middleware to enhance collaboration in various activities.

Conventionally, middleware is a distributed software layer, which abstracts over the complexity and heterogeneity of the underlying distributed environment with its multitude of network technologies, machine architectures, operating systems and programming languages (Muruganatham et al 2010). The research project purpose was to extend distributed computing to mobile phones utilising android environment. The android mobile operating system is open source, with a software stack for mobile devices, thus providing rich environment for grid middleware development.

Implementing a middleware for mobile grid computing on android enhanced collaborating collection of mobile devices to optimally use available resources. The middleware provided a layer of abstraction that enabled mobile phones on a grid act as collaborating node, extending grid computing to mobile devices. Distributed middleware system that can achieve such collaboration would improve the user experience by providing abstraction layer for process migration, mobility and resource sharing.

1.5 Objectives

1.5.1 Overall objectives

The overall objective of this project was to develop a mobile phone middleware framework for grid computing that support resource sharing and mobility on the android mobile operating system.

1.5.2 The sub objectives

To review android mobile phone operating system platform.

- i. To implement mobile grid middleware on android platform.
- ii. To demonstrate applicability of grid computing in mobile environment using the developed middleware.

1.6 Research Question

The use of mobile phones in grid environment poses the significant question which form part of the research scope: How to implement mobile phone grid middleware on android platform to harness the raw power of resources that lies idle when the phone is not in use?

1.7 Assumptions and limitations of the research

The following assumption were considered on this project

- i. Android provides open source mobile platform which is secure and fault tolerant to be used as grid operation system of choice
- ii. Mobile phone users will accept and permit use their personalised phones on the grid environment to allow collaboration
- iii. Phones used on grid support WIFI and each phone act as node on the network. WIFI will be used as mode of communication among the nodes on the grid
- iv. Each phone will represent a grid node and will have the middleware installed
- v. The various context information of the phone that is processor utilisation level and battery level have same priority
- vi. The nodes are in the same geographical solution where WIFI communication is available among the nodes

The major limitation involves addressing various challenges and constraints experience in early initiatives to develop a mobile phone grid environment. These challenges include the following; Mobility, where location is no longer fixed, reliance on battery power and high levels of operating heterogeneity among mobile phones in terms of hardware and software;

1.8 Organization/ Structure of the Report

In the remainder of the report, Chapter 2 provides a background to the relevant literature on mobile grid computing and middleware technologies, looking at the history, development and weaknesses that have become apparent. It provides a description; summary, critical evaluations of each related previous work done and review of the android mobile operating system.

Chapter 3 gives information on the methodology used the middleware is design in order to meet the functional requirements. The chapter presents the analysis of the deferent requirements for this project including functional requirements and non-functional requirements. The software and hardware requirements, communication framework and operation are highlighted in order to successfully implement middleware.

Chapter 4 presents how the middleware is designed in order to meet the functional requirements. The chapter gives the overall middleware architecture, module design and diagrams relating to project operation.

Chapter 5 explains how the implementation of each middleware was performed after the overall design in previous chapter. The Focus included, setting up development environment and how the various modules are implemented.

Chapter 6 describes the tests that were carried out on the elements of the middleware. It was shows how the testing of the overall project was implemented and results achieved.

Chapter 7 presents achievements, challenges, brief description of the potential future works and conclusion of the work done in this research project.

CHAPTER 2: LITERATURE REVIEW

2.0 Grid Computing Overview

Grid computing is a significant ongoing computing initiative that involves the aggregation of interconnected nodes of computing devices to form a large-scale, distributed system for coordinated problem solving and resource sharing. The Grid allows the coupling of geographically distributed resources to offer consistent and inexpensive access to resources irrespective of their physical location or access point. The Computing tasks forms a workload that is subdivided in sub tasks distributed among the member nodes. This enables the grid users to benefit from enormous computation, storage, and bandwidth resources that are not readily available in mobile devices.

Grid Computing, at its core, enables devices to be virtually shared, managed and accessed across the collaborating nodes, regardless of their operating system characteristics. Marvi, gives analogy that grid computing is similar to power grids, where a user does not need to know anything about what stays beyond the socket, (Manvi and Birje, 2008). This indicates importance of a middleware to give users the needed abstraction on grid environment

Grid computing enables the ability to distribute jobs to collaborating computing devices by having smaller subtasks of the load that are distributed the evenly based on resources and policies, (Advanced technology explored 2011). Other advantages of the grid computing include the following; Ability to solve larger, more complex problems in a shorter time, easier to collaborate with other nodes, Make better use of existing hardware, software and other computing resources, Allow widely dispersed computing resources to create virtual infrastructure to share data and resources , create flexible, resilient and fault tolerant operational infrastructures. However, there are few disadvantages of grid computing which include; non-interactive job submission, due to the lack of central control over the hardware, there is no way to guarantee that nodes will not drop out of the network at random times, communication overheads and the fact that resources exist in different administrative domains, run different software, and are subject to different policies.

2.1 Characterization of Distributed Systems

Distributed system consists of a collection of components, distributed over various computers (also called hosts) connected via a computer network (Licia et al, 2001). The hosts require interacting with others via a communication network. Although this interaction may be built directly on top of network operating system primitives, this would be too complex for many application developers, thus a middleware which is layered between distributed systems. The figure 2 illustrates an example of distributed system.

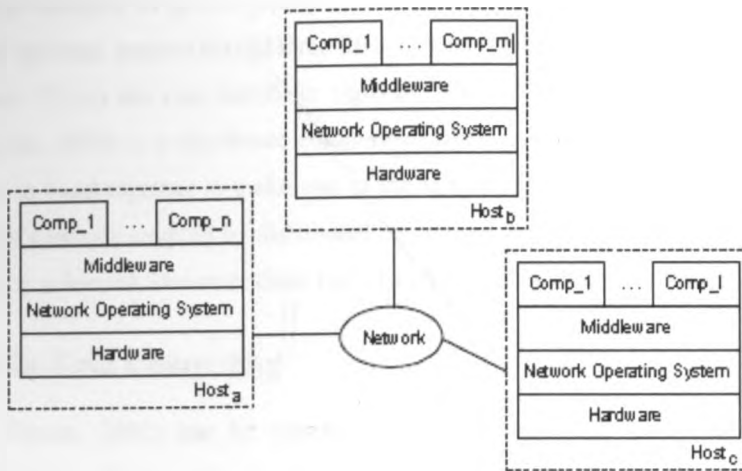


Figure 2: Example of a distributed system (Licia et al 2001 p4).

In addition, (Licia et al 2001), suggests that the concepts of device, of network connection and of execution context, greatly influences the type of middleware used. These results to various characterizations of mobile distributed systems as depicted in the figure 3.

Distributed System

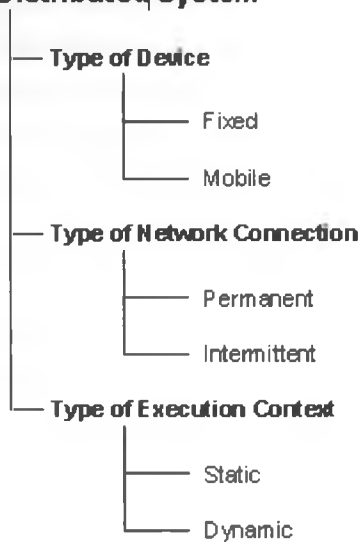


Figure 3: Characterizations of mobile distributed systems

In the review of wireless grid architectures, can be broadly categorized into four classes based on the devices predominant in the grid and the relative mobility of the devices in the grid (Manvi and Birje 2001). The categories include fixed wireless grids, mobile or dynamic wireless grids, Ad Hoc and grids and sensor network grids.

A well-known example of grid computing in the public domain is the ongoing SETI (Search for Extraterrestrial Intelligence) @Home project (Seti@Home 2010) in which thousands of people are sharing the unused processor cycles of their PCs in the vast search for signs of "rational" signals from outer space. For instance Folding home (Folding@home 2010) is a distributed computing project, where people from throughout the world download and run software to band together to make one of the largest supercomputers in the world. Folding@home uses novel computational methods coupled by distributed computing, to simulate problems millions of times more challenging than previously achieved. However these two examples rely on traditional fixed grid with centralized administration.

2.2 Mobile Grid Computing

The Grid, (Foster, 2002) can be viewed as a distributed, high performance computing and data handling infrastructure, that incorporates geographically and organizationally dispersed, heterogeneous resources and provides common interfaces for all these resources, using standard, open, general-purpose protocols and interfaces. Antonios 2007, defines, mobile Grid as a platform that should address mobility issues by means of enabling both fixed and mobile users to have access to both fixed and mobile grid resources utilizing transparently and efficiently the underlying technologies. This type of grid involves user mobility and terminal mobility, where the user and the nodes are not fixed, thus can change location.

The advances in Mobile devices promise ubiquitous computing. This has led to many researchers further venture onto grid computing, which over the years has evolved into a de facto standard for shared resource computations. There are several approaches to attempts to bridge the disparity between mobile and desktop experience in computing. Capra argues that recent advances in wireless networking technologies and the growing success of mobile computing devices, such as laptop computers, third generation mobile phones, personal digital assistants, watches and the like, are enabling new classes of applications that present challenging problems to designers (Capra et al, 2002). However these devices face limitations of irregular connectivity and scarce resources, thus the research in the field of middleware systems has proliferated.

2.3 Previous work done on Mobile Middleware

Junseok and Praveen, proposed middleware architecture, called Scalable Inter-Grid Network Adaptation Layers (Signal), integrates mobile devices with existing grid platforms to conduct peer-to-peer operations through proxy-based systems (Junseok and Praveen, 2004,). Combining mobile P2P applications with grid technologies could ultimately give mobile devices the power of supercomputers. The implementation follows OGSA's service description model, which details how grid service contains the service data elements (SDEs) that define service parameters.

The Fluctuation of resources on a single device has resulted to attempts to scale up mobile experience. David and Marty, implements mobile OGSINET to address the mobile device resource limitations and intermittent connectivity. The mobile OGSINET is an important step towards making mobile a first class entity in grids based on OGSINET (Open Grid Service Infrastructure), (David and Marty, 2004). Though, the OGSINET implementations exists on

various platforms or runtimes for instance Java virtual Machine. The mobile OGSINET runs only on Microsoft Pocket PC operating system only which is proprietary and this implementation does not fully address mobile devices constraints.

Umar presents architecture of a middleware layer that enables users of mobile devices to seamlessly and securely access distributed resources in a Grid (Umar et al, 2005). The middleware architecture that addresses the issues of job delegation to a Grid service, support for online processing, secure communication, interaction with heterogeneous mobile devices and presentation of results formatted in accordance with the device specification. Conversely, it only seeks to extend potential of the traditional fixed Grid to a wider audience, particularly for users of mobile devices.

Licia argues that middleware solutions for wired distributed systems cannot be used in a mobile setting, as the principle of transparency that has driven their design runs counter to the new degrees of awareness imposed by mobility (Licia et al, 2001). They propose a synergy of reflection and code mobility as a means for middleware to give applications the desired level of flexibility to react to changes happening in the environment, including those that have not necessarily been foreseen by middleware designers. The designers, when developing distributed applications do not have to deal explicitly with problems related to distribution, such as heterogeneity, scalability, resource sharing, and the like. Middleware developed upon network operating systems provides application designers with a higher level of abstraction, hiding the complexity introduced by distribution.

The Internet's TCP/IP is the de facto global data and communication protocol. Built on top of TCP/IP, web services standardize a means to programmatically access remote procedures to offer application interoperability. The Open Grid Services Infrastructure (OGSI), a normative specification, quickly followed. Collectively these define grid services, extensions to the SOAP communications protocol for grid computing. This provides true platform-independent grid computing.

Victor, implements a distributed mobile application framework based on Nokia's Symbian operating system (Victor, 2009). The distributed communications framework is aimed to provide a good way for the parts of a distributed system to communication on mobile environment.

Masinde, proposes Mobigrid, which is an API on which distributed mobile application is developed to extend grid computing to mobile grid (Masinde et al, 2010). Further, Masinde, implements Mobigrid, a Distributed Mobile Application Framework. The middleware uniqueness lay in the fact that the middleware is for mobile phones environment, purpose to bridge the technological gap that exists in the rural areas of the developing countries of Africa where the adoption of mobile phones technology is higher than that of other forms of ICTs. MobiGrid was initially implemented using Python programming language for S60 (pyS60) and tested on only two phone models: the Nokia E63 and Nokia N95. However, MobiGrid had several limitations two of the major ones being;

- i. MobiGrid could only recognize coordinators using the subnet mask 255.255.255.0, due to the lack of support for broadcast in PyS60, requiring use of a work-around implementation

- ii. The MobiGrid required that the two main modules (Local Server and Coordinator Server) be installed on each of the phones participating in the grid to function properly. Thus, due to limitations of S60 (at the time of initial developing the application), a phone could not run more than one instance of the application hence impossible to come up with a custom application to test MobiGrid. Android allows multiple instances to run as background services.

Jung, proposes Message oriented middleware (MOM) is a specific class of middleware that operates on the principles of message passing or message queuing (Jung et al, 1999). This complements de facto message passing techniques like Local procedure calls (LPC) and Remote Procedure Call (RPC)

The communication will be based on WIFI; many smart-phones provide a wide number of connectivity options including WIFI and 3G Internet connectivity. WIFI offers zero cost communication when used for peer-to-peer networking. Wireless networks are inherently broadcast, however early forms of wireless security, such as WEP, have proven inadequate. The infrastructure is adopting more secure protocols, such as WPA. This has resulted to security being a foundation design principal rather than an afterthought.

Other previous related work suggests several approaches to address this problem. David et al, summarizes this approaches as:

- i. Mobile collaborative computing tools
- ii. Single-device resource management;
- iii. Multi-device grid computing resource management.

2.4 Android Mobile OS

Android is a software stack for mobile devices that includes an operating system, middleware and key applications developed by Open Handset Alliance (Android Developers 2011). It relies on a Linux 2.6 kernel for core system functionality and runs code written in the Java programming language on a specially designed virtual machine named Dalvik. Dalvik executes Dalvik Executable files which are Java compiled classes optimized to minimize memory footprint. The overall system architecture is found in figure 4.

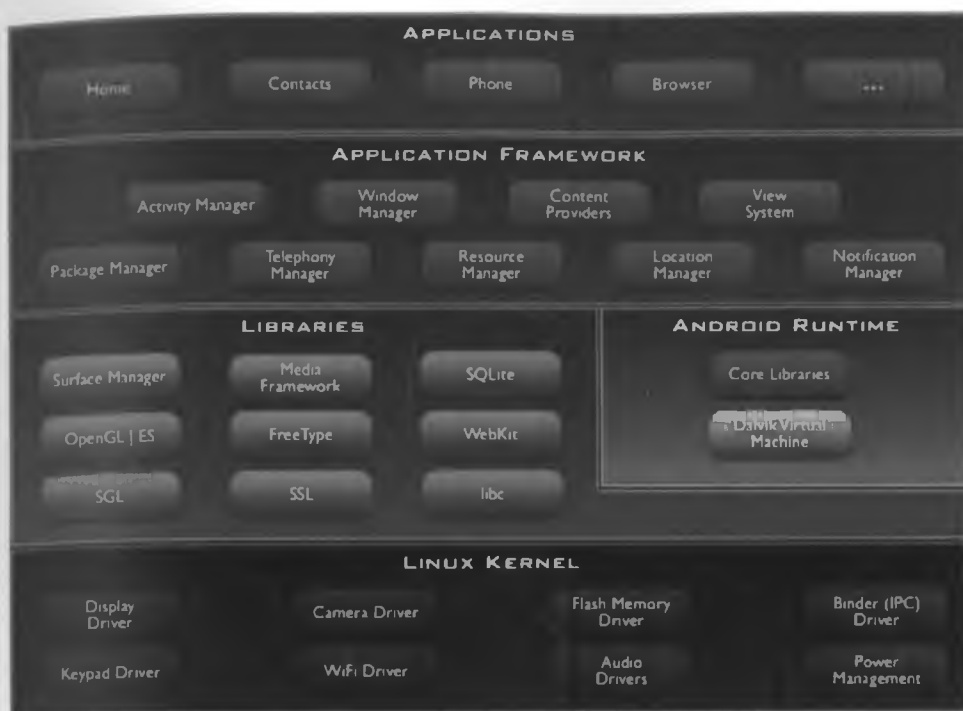


Figure 4: Android Architecture

A detailed description of android stack system architecture is given below

Kernel: - The base of the Android stack is the Linux kernel. Android uses Linux as a hardware abstraction layer and use the powerful of Linux kernel. This kernel acts as the layer between the mobile-device hardware and the rest of the Android stack. It manages memory, processes, file systems, and all I/O operations.

Android runtime: - On top of the Linux kernel is the Android runtime. Android includes a virtual machine called Dalvik that runs Java applications. It is a version of Virtual machine optimized for mobile devices to run with a low memory footprint and optimized hardware resource. Every Android application runs in its own process within a Dalvik virtual machine. The application is packaged into a Dalvik executable file that the Dalvik virtual.

Libraries: - Android stack includes two sets of libraries. This includes core libraries that provide different functionality to the applications that run on Android. The second set is native libraries developed using the C and C++ languages preinstalled by vendor in most cases for instance surface manager, 2d and 3d graphics, media codec and sql lite database.

Application Framework: -The C and C++ libraries in the Android stack are exposed to developers and other applications through the Application Framework. The Application Framework also enables applications to register functionality that the other applications can reuse.

Applications:-The Android stack ships with a basic set of applications. This application are developed using java language utilizing android SDK. Examples of application include an email client, short message service (SMS)

client, calendar, browser, and contacts application. In this project an application is developed to demonstrate applicability of the middleware.

The Android has SDK provides the tools and APIs necessary to begin developing applications on the Android platform using the Java programming language This includes both an Eclipse IDE plug-in, emulator, debugging tools, visual layout builder, log monitor and more. Various features of the android mobile operating system will be reviewed in details later in this project research

2.5 Why Android?

In contrast to most other mobile platforms, Android is available as open source software. This enables devices to be customized without restrictions and enables any device manufacturer to ship devices with Android. Likewise are developers able to distribute applications to any android device trough the Android market. Unlike Apple's iPhone platform, application distribution does not require any external review or acceptance and multiple application markets exist.

Android provides access to a wide range of useful libraries and tools that can be used to build rich applications (Android 2011). For example, Android enables developers to obtain the location of the device, and allows devices to communicate with one another enabling rich peer-to-peer social applications. In addition, Android includes a full set of tools that have been built from the ground up alongside the platform providing developers with high productivity and deep insight into their applications.

Within the context of this research project, Android is especially interesting due to its state of the art status. It is a newly released platform which has already gained massive support from device manufacturers Open Handset Alliance partners. This is expected to make it an attractive platform, since the development of a single application can reach a broad range of devices, all with a rich user interface experience. Gartner reported that Android as an operating system had greatest growth percentage (888.8%) in 2010 and is now in the number two position worldwide.

**Worldwide Smartphone Sales to End Users by Operating System in 2010
(Thousands of Units)**

Company	2010 Units	2010 Market Share (%)	2009 Units	2009 Market Share (%)
Symbian	111,576.7	37.6	80,878.3	46.9
Android	67,224.5	22.7	6,798.4	3.9
Research In Motion	47,451.6	16.0	34,346.6	19.9
iOS	46,598.3	15.7	24,889.7	14.4
Microsoft	12,378.2	4.2	15,031.0	8.7
Other Oss	11417.4	3.8	10432.1	6.1
Total	296,646.6	100.0	172,376.1	100.0

Source: Gartner (February 2011)

Table 1: Android market growth

2.6 Conclusion

In the Challenges and Lessons in developing Middleware on Smart Phones, (Oriana R, 2008), concludes that the most useful feature of smart phones as enablers of pervasive computing is the possibility of installing new applications, which in many cases anyone can write. It is in this spirit that the middleware will be developed

CHAPTER 3: METHODOLOGY

3.0 Chapter Overview

This chapter presents the methodology used in the middleware design in order to meet the functional requirements. The following sections analyze the deferent requirements for this project including functional requirements and non-functional requirements. The software and hardware requirements are also listed in order to successfully implement the project.

3.1 Methodology Overview

The middleware development process was geared towards ensuring transparency and timely delivery of a functional middleware. The process focused on problem statement, literature review, android mobile platform review, system design, overall system implementation and testing.

Prototyping was highly deployed. Operational prototyping was used to build and test progressively the individual features of the middleware. Once a feature was fully tested, they were added to the operational prototype and further evolved.

The implementation approach followed the software development lifecycle from requirement specification to testing and deployment. However, the iterative agile methodology was used since it enables one to take advantage of what was being learned during the development of earlier, incremental, deliverable versions of the middleware framework.

3.1.1 Agile Methodology

Agile methodology is a software development methodology typically used in rapid development and implementation. Gail sums up that it is designed to deliver projects through incremental, iterative work cadences known as sprints (Gail, 2010). Further, the iterative approach enables the stakeholder and the developer to have frequent interaction to review the direction of the project, and if necessary, make revisions early to avoid costly changes that would otherwise not be identified until much later in the waterfall methodology. This result to “inspect-and-adapt” approach to development greatly reduces both development costs and time. The output of each iteration was a working code that was used to evaluate and respond to changing and evolving user requirements.

3.1.2 Limitation of the Methodology

Grid computing being relatively new in mobile phones the following challenges were expected in the course of project development

- i. Access/availability to previous work done
- ii. Learning android Mobile operating system

- iii. Limited time
- iv. Prototyping technique proposed is time consuming.
- v. Agile development methods do not scale. Due to the integrative approach, it is hard for some to understand exactly where the project stands.

However, these challenges were mitigated by the fact that the literature review was widely done, through consultation with the supervisor; tutor based android learning and doing various activities concurrently to save on time. Iterative agile methodology was used to bridge the gaps noticed since it enables one to take advantage of what is learnt during the development process.

3.2 Requirement Specification and Analysis

System requirements should set out what the system should do rather than how this is done. A requirement may be a functional requirement, describing a system service or function. Alternatively, it may be a non-functional requirement. The requirements for the middleware were defined as follows.

3.3 Middleware Functional Requirements

This requirement defines functions of a middleware system or its components, describing an activity or process that the system must perform. The middleware should be able to provide the following key functional requirements

- i. Node discovery: The client should be able to connect to server node. The server should be able to listen for client connection via sockets on a defined port.
- ii. Resource discovery, the node should be able to return its resources like battery level, memory among others. Once connected the server should be able to get the ip address of the client and the client node resources
- iii. Status;- server / coordinator node ability to give the status of clients nodes
- iv. Ability to communicate with other nodes via WIFI
- v. Communications via sockets
- vi. Ability to collaborate in performing a task carried on the test application
- vii. Act as grid API, that support an application for demonstration purposes

The diagrams 5 illustrates a sample Use Case class diagram of the client Middleware node

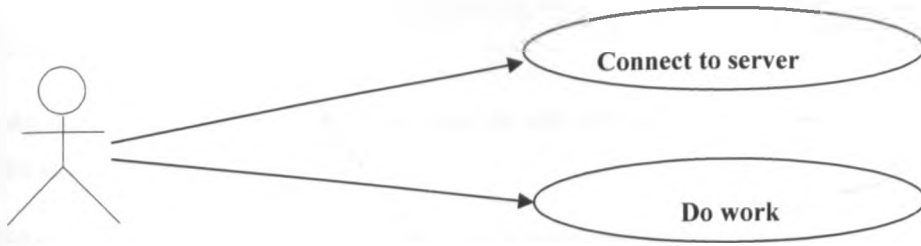


Figure 5: Use Case Description

Use Case Name	Client Node Connection to Grid Middleware
Primary Actor	User
Data	Computation task
Description	The user starts the application.
Preconditions	None
Post Conditions	The application connects to server via sockets and does work

Table 2: Use case for Do work

3.4 Non Functional requirements

A non-functional requirement is a requirement that specifies criteria and qualities that can be used to judge the operation of a system, rather than specific behaviours.

The system was required to meet the following non-functional requirements:

- i. Compatibility with various version releases of android mobile OS
- ii. Flexible messaging system, allowing the developer using the framework to determine what parameters to be passed when calling a service
- iii. Performance – The middleware must load with ease, warn the user if there is any error. Ability to keep logs.

- iv. Extensibility – The software should be written in such a way as to allow extra functionality to be added in the future;
- v. Maintainability – The middleware code must be well-structured, readable and well-commented to allow for the implementation of extra functionality;
- vi. Efficiency – The software should use as little resources as possible to perform at an acceptable level;
- vii. Reliability – The system should load at any time and provide reliable information. Any errors must be handled with minimal hindrance to the user;

3.5 Hardware and Software Requirements

Hardware Equipments:

- Laptop:
- Two android powered smart phones (Huawei Ideos). The phone should be running Android OS version 2.2 code named (Froyo) or greater. The phone must be running a chip based on the ARM v6 instruction set

Software:

- Android SDK for windows
- Java JRE 6
- Android Virtual Device (AVD) plug-in for Eclipse
- Eclipse IDE for Java Developers Version: Helios Service Release 1

3.6 Communication Framework

3.6.1 Socket Communication

The communication between nodes relies on socket, which is an endpoint for communication between two machines. In this regard TCP is used protocol, simply because it is used for so many applications such as HTTP, POP, SMTP, etc. and guarantees that the receiver will receive exactly what the sender sent with minimal errors.

The most common way to make communication between distributed applications is by using operations on sockets. The basic principle is explicitly exchanging messages using send and receives command of the socket mechanisms. The message itself can be encoded in a binary or text form. For example, a client may issue a REQUEST command to a server using TCP protocol. The server then sends back RESPOND command to the client using the same protocol.

The diagram how the communication takes place between the nodes via TCP sockets

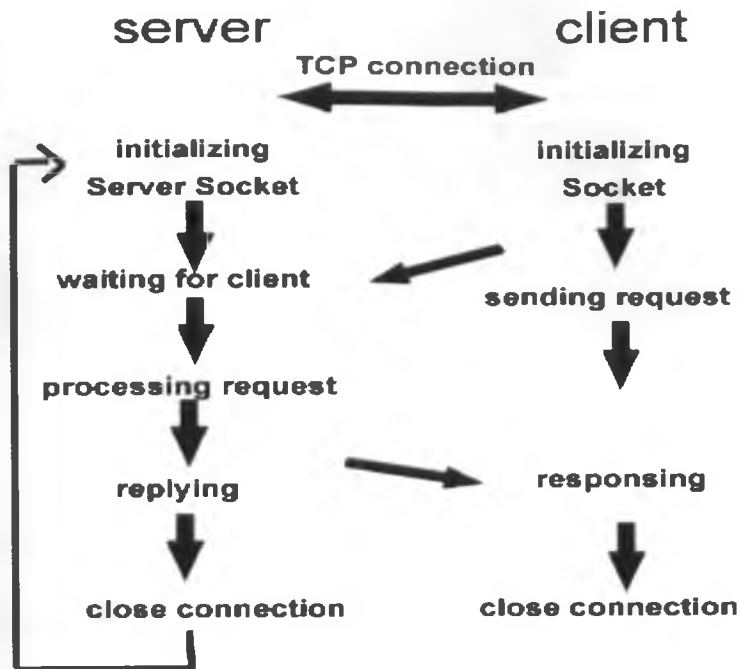


Figure 5: Nodes TCP communication time diagram

The node runs processes that facilitate communicating via TCP sockets. Each side of a TCP connection has a socket which can be identified by the pair $\langle IP_address, port_number \rangle$. Two processes communicating over TCP form a logical connection that is uniquely identifiable by the two sockets involved, that is by the combination $\langle local_IP_address, local_port, remote_IP_address, remote_port \rangle$.

The communication is layered from the android application (sender node) through other layers, until it's delivered to the receiver node. Data communication in this project involves create a Socket object, connect to remote host, send data and wait for data to come back

This is illustrated in the figure 6.

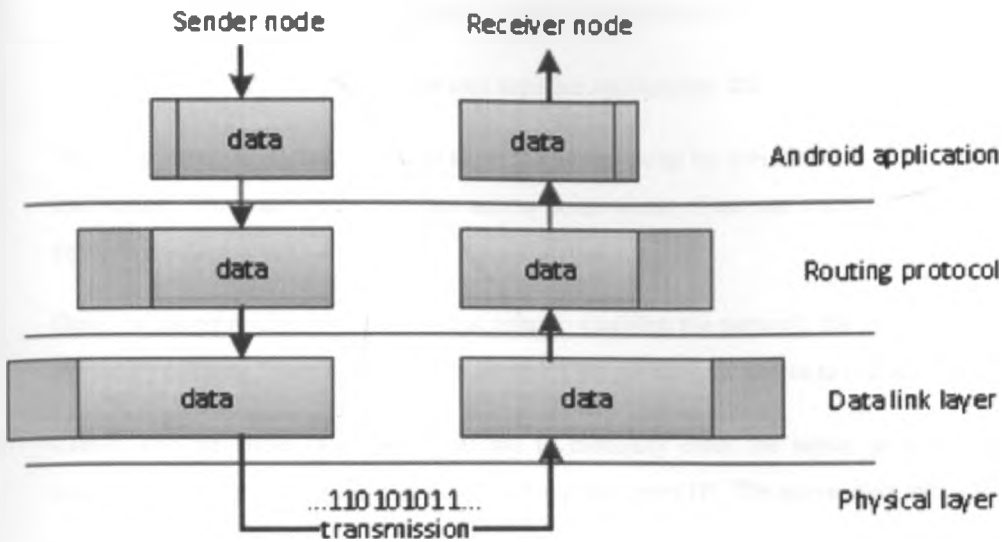


Figure 6: Packet Data Units (PDU) Packaging Layer

3.6.2 Peer to Peer communication

Android operating system does not support peer to peer WIFI based communication between phones. Thus for phones to communicate, an access point was setup. Most android smart phones are rooted, meaning you can access SU user (super user) privileges. How over to access the phones root one has to hack the phone which means to write a patch for the operating system). In ad-hoc mode there is no routers, only the mobile phones connected to the ad-hoc network, so in order to achieve communication between nodes, socket communication was chosen.

3.7 Operation of the middleware

The middleware was designed to work by providing a middleware service that is always started and runs in the background to allow distributed applications to operate. This middleware service abstracts the developer of the distributed application from having to deal with the technicality of locating distributed services and managing remote nodes. The typical operation of Middleware is as follows:

Grid computing involves the aggregation of interconnected nodes of computing devices to form a large-scale, distributed system for coordinated problem solving and resource sharing.

The task handling in a grid is pretty much straight forward; it's broken into smaller computable subtasks which are sent to participating nodes for computation and then the returned results is merged. This is facilitated by the grid middleware developed.

The detailed operation process is illustrated on the architecture figure and activity sequence diagram figure

- i. Initialization, where the user starts the Middleware service on his/her phone by selecting it from the menu.

- ii. The assumption here is that Middleware is already installed on his/her phone.
- iii. The middleware implementation has two separate applications that are client and server.
- iv. When the server is started, you need to set the access point by initializing WIFI and android WIFI tethering application. This establishes wireless access point where client can connect. Clients were assumed WIFI enabled
- v. Once the server or client is set as access point to establish the network, the middleware communication was started by clicking "Start button on the server" This prompts the server to return and display its local ip.
- vi. Client: For the client to connect you has to manually enter the server ip and click "connect". Once connected the client indicates connected and returns server IP: The server also indicates client connected.
- vii. Alternatively if WIFI is not set the server to establish local WLAN, the server will return public IP it is connected on the internet and if this ip is put on client the connection can be established regardless of the client or server location since 3G/ GPRS mobile provider network will be used.
- viii. Once communication is established, then we can embark into business of running an application of choice.
- ix. However the system resource dictates if the phone will participate as a client or server node. For instance if battery level is less than 25 % the connection will be terminated and user advised to charge the phone
- x. Middleware provides an abstraction layer by having libraries that query the nodes system resources. These resources are displayed on the middleware Application GUI on the system resources tab.
- xi. The middleware server / coordinator keeps track of all node and the interactions with the server

3.8 Conclusion

The requirements presented here provide the basis for the design of the system, they provide what will be expected of the system not only by the user but by what a potential future users may require.

CHAPTER 4: SYSTEM DESIGN PHASE

4.0 Chapter Overview

This chapter presents how the middleware was designed in order to meet the functional requirements. The following sections analyze the overall middleware architecture, module design and diagrams relating to project operation.

4.1 Middleware Architecture Overview

The system design involved various modules that helped to achieve the overall framework. The communication was via TCP socket message passing utilizing WIFI.

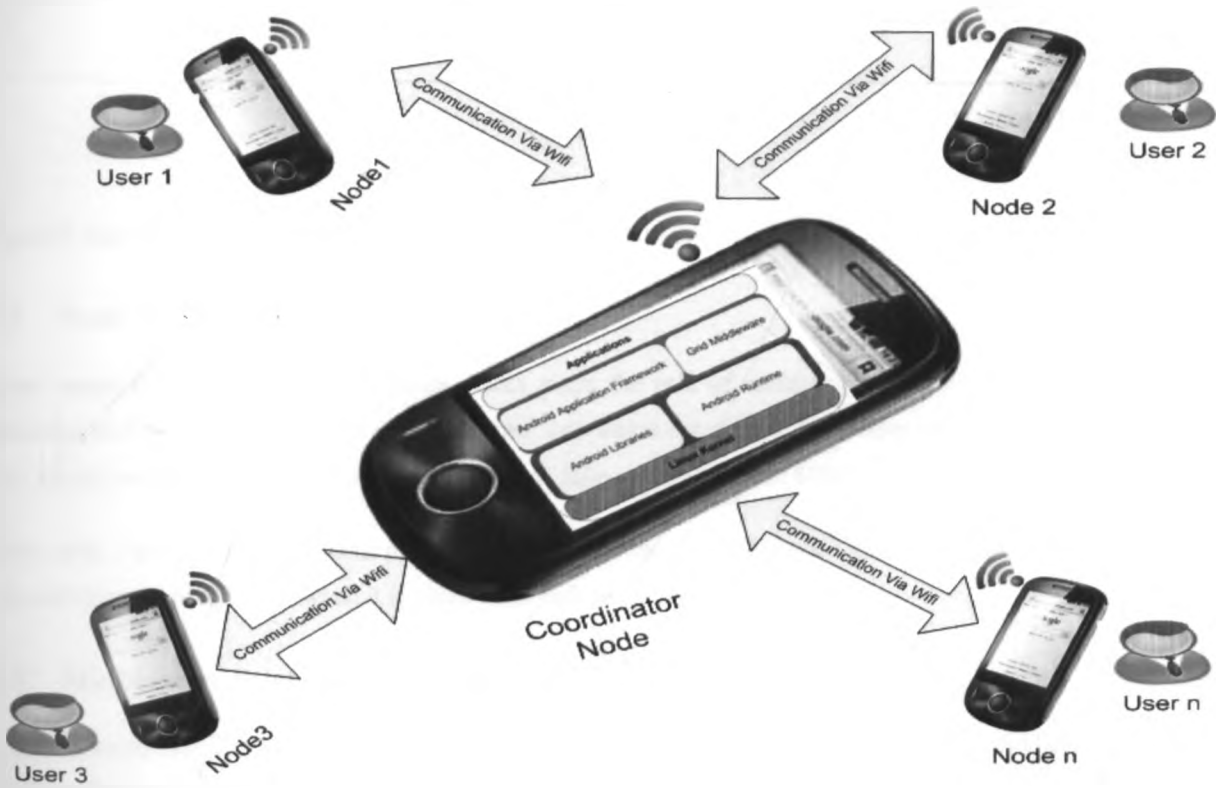


Figure 7: Proposed middleware architecture

4.1.1 Proposed System Detailed Design

The detailed design of the proposed middleware System is illustrated in Figure 8:

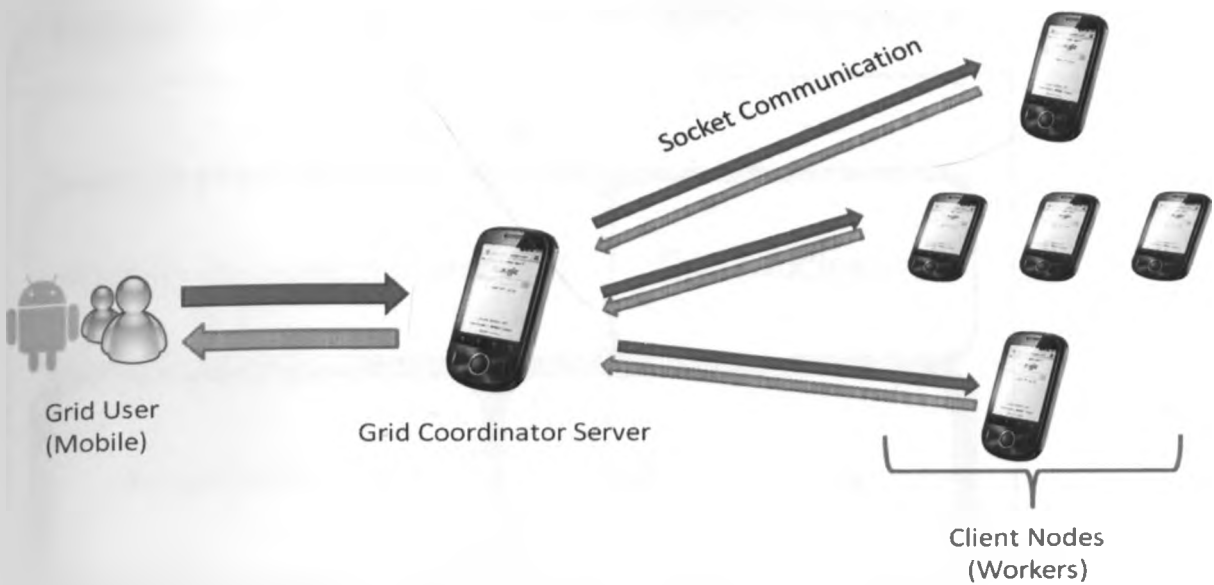


Figure 8: Detailed design of the middleware System

4.2 Node Functions

Sever node: this is the coordinator node which plays the role of managing other nodes communication, task allocation and result aggregation. It acts as a server, and listens for clients' connection on its IP address in a specific port. In this project port 8080 is used, the syntax is as follows < server IP>: 8080

User node: This refers to the client phone nodes that join the grid. This node takes a role of either a client when submitting a task or a worker when it is given some task to compute.

4.3 Middleware integrated with android software stack

The framework design is based on android overall system architecture. This is to enable the middleware achieve its functionality of providing the abstraction to users. The figure 9 shows middleware integrated with android framework.

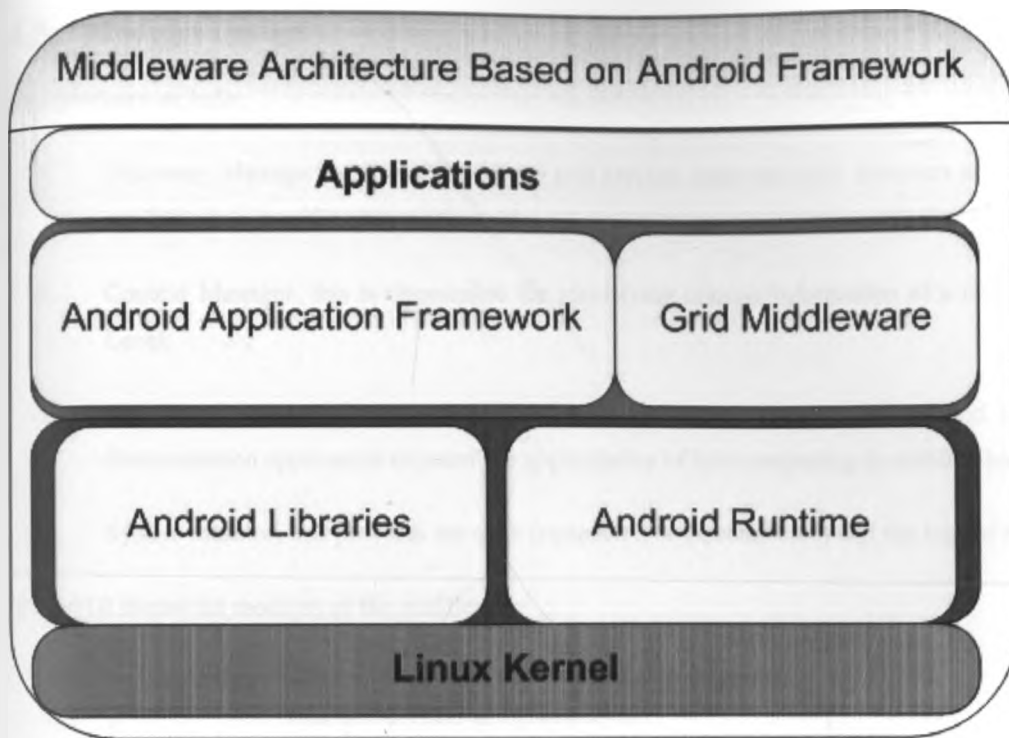


Figure 9: The diagram Android Software Stack

4.4 Android Applications

In this project, Android application was developed, the application composed of four basic building blocks: activities, services, content providers and intent broadcasters. These components were bundled together with the resource files the application requires to form the application. The final bundled file was compressed to an APK file. Each APK file is considered one application and is used to distribute and load the application onto the phone. Because each component can be called individually the application is started when any component of the application needs to be executed, and shuts down when no longer needed or system resources are required by other applications

Permissions can be set to share data between processes and applications can call one another using intents. Permissions used in this project are discussed on implementation chapter. The use of intents and broadcast receivers was highly used to provide extended functionality in response to an application call by adding functionality to the call or replacing the functionality altogether. Activities present the graphical display of to the application. They provide a user interface and react to user inputs or events. Although activities are independent, they work together and communicate to make one cohesive user interface.

4.5 Modules Design

The Modules include:

- i. Discovery Manager, which initializes the grid service, scans the grid, discovers and advertises the resources available in a specific node on the grid.
- ii. Context Manager, this is responsible for identifying context information of a node like CPU and Battery Level.
- iii. Activities Manager, this coordinates activities by monitoring applications' and system calls. It has the demonstration application to proof the applicability of grid computing in mobile phones
- iv. System monitor, this provides the state (connected or disconnected) and the logs of a node on the grid

Figure10 shows the modules of the middleware.

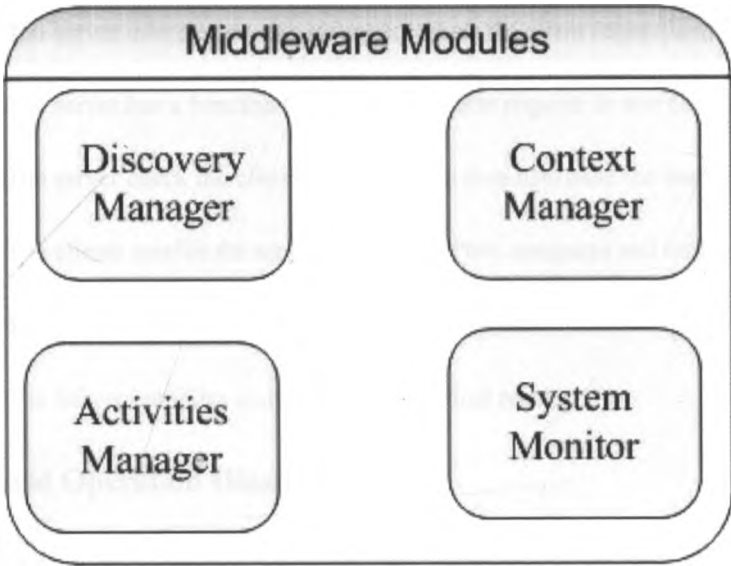


Figure 10: Modules of the middleware.

4.6 Test Case Scenario

To demonstrate the applicability of the grid middleware, a simple application for number summation was used. Getting sums of many numbers is a memory bound calculations that involve many additions of quite big numbers, thus they take a long time (not only for doing the operation but also to handle the memory). The application adds a determined size numbers. The number, of numbers to be added was determined by the user during the grid application initialization as a client work request. The server or the coordinator generates a workload based on request received utilizing the inbuilt function for task generation, in this case a random number generator. The task of adding numbers generated was distributed to available work nodes or clients that were connected to the system

for computation. Each node then added the numbers that had been sent and return the sum to the server. The server merged all subtask sums to get the final sum. The final result was sent to the requesting client.

This acts as an analogy of an application like gene string comparison, drought prediction, harvest prediction among others distributed which are computing resource intensive and may need the collaborating participants submit requests via mobile and get instant results

4.6.1 Test Case process:

The following illustrate the process of the test case used.

- i. The server and client were installed with respective middleware applications. The server was initialized and clients connected.
- ii. Client sends Request to the server to generate some random numbers by sending the size of the random numbers to be generated
- iii. The server middleware application. receives the client request and process it
- iv. The server has a function to process the client request. In this case the random numbers are generated
- v. The server check the clients connected , It then distribute the load to the clients
- vi. The clients receive the assigned subtask. Then computes and returns the results of the subtask
- vii. The client sends subtask results back to the server
- viii. The Server compiles and sends back the final results

4.7 Grid Operation Illustration

The Middle operation from generating a task to distributing subtasks to clients as per the process above is illustrated in figure 11.

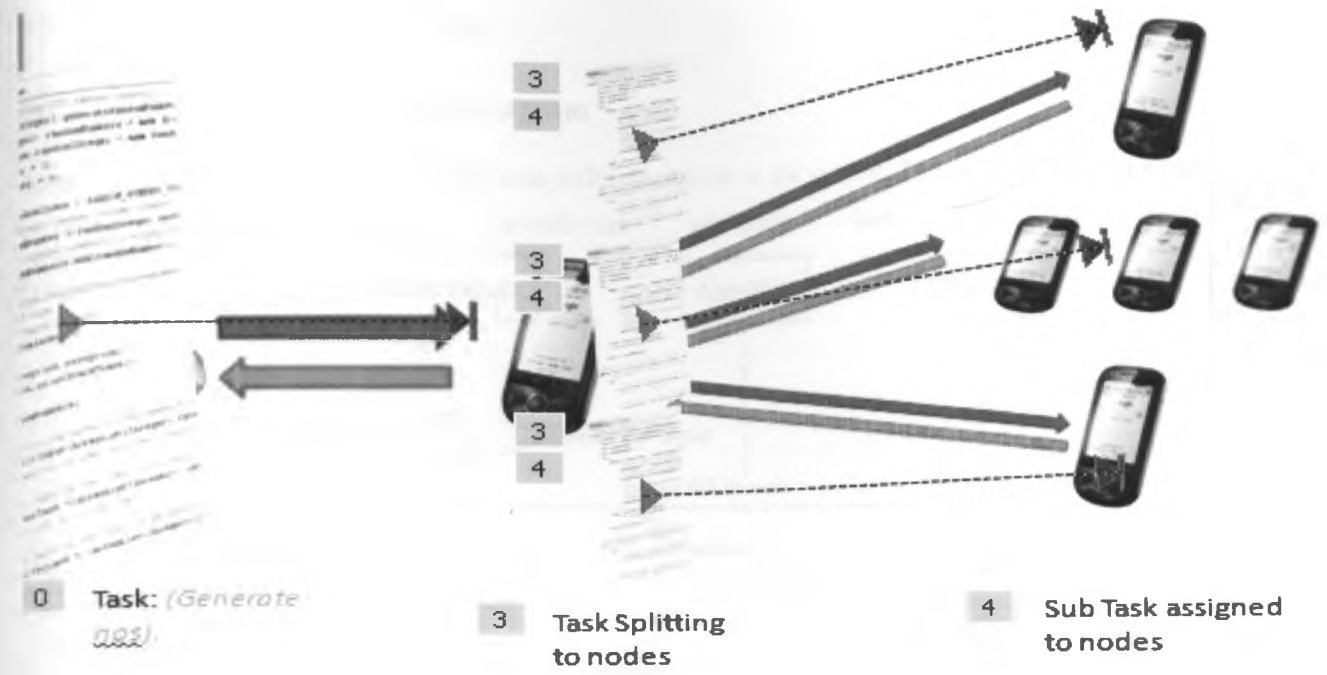


Figure 11: Task Generation illustration diagram

4.8 System High Level Diagrams

4.8.1 Activity Sequence Design Diagram

Sequence diagram shows the messaging between different objects in the system. The diagram below is a high-level sequence diagram of the middleware operation and the test application.

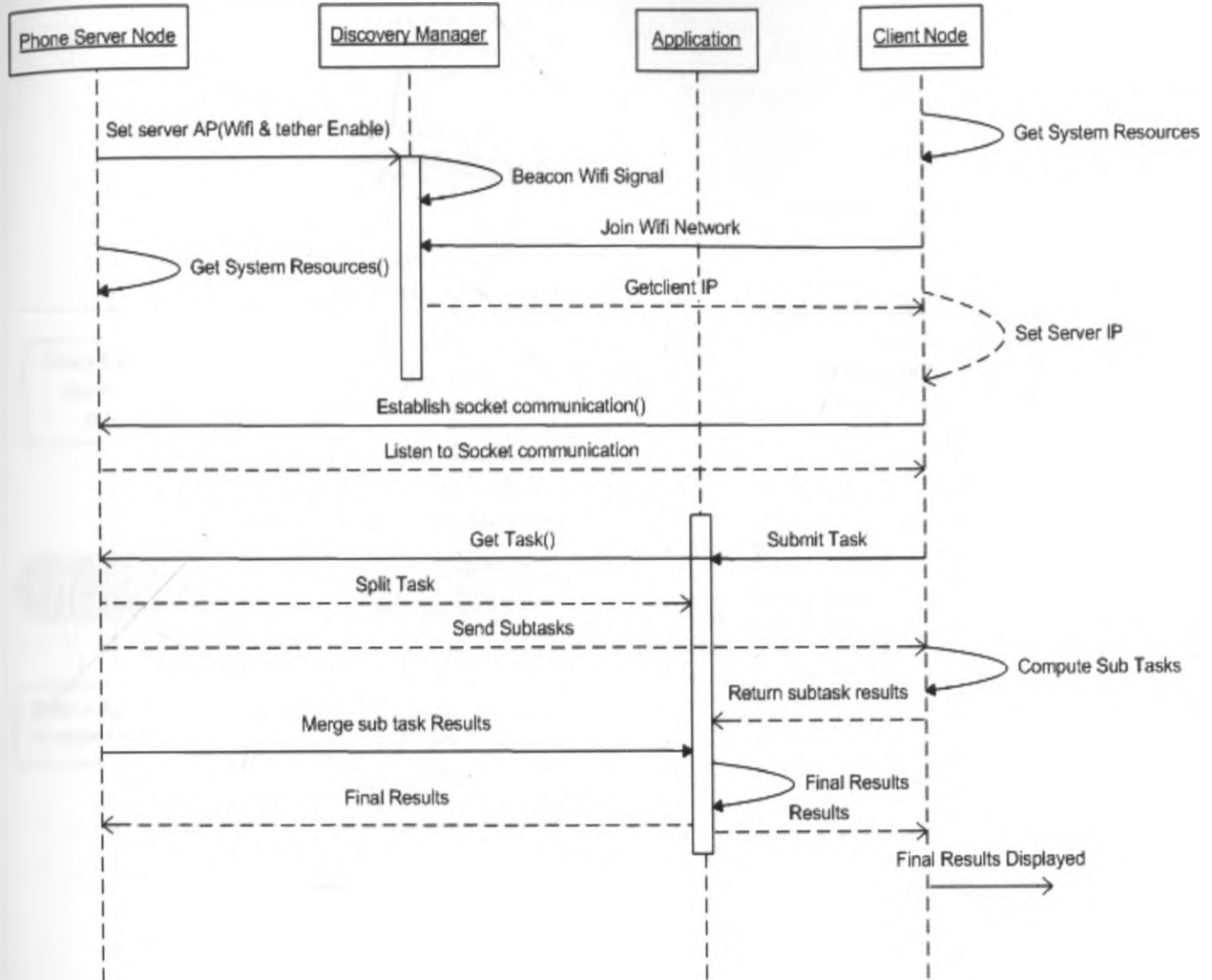


Figure 12 : Middleware Sequence Diagram

4.8.2 Android Activity Diagram

The implementation of the project was based on android mobile operating system. The android OS implementation is based on packages, then classes with various methods which deploy use of activities. The activity is a single, focused thing that the user can do..Almost all activities interact with the user, so the Activity class takes care of creating a window for you in which you can place your UI with setContentView (View) Method. Various methods

in almost all subclasses of Activity will implement, onCreate (Bundle) is where you initialize your activity and onPause() where you deal with the user leaving your activity.

The Diagram below illustrates the lifecycle diagram on the android activities used in the middleware development. The colored nodes indicates the user interaction points

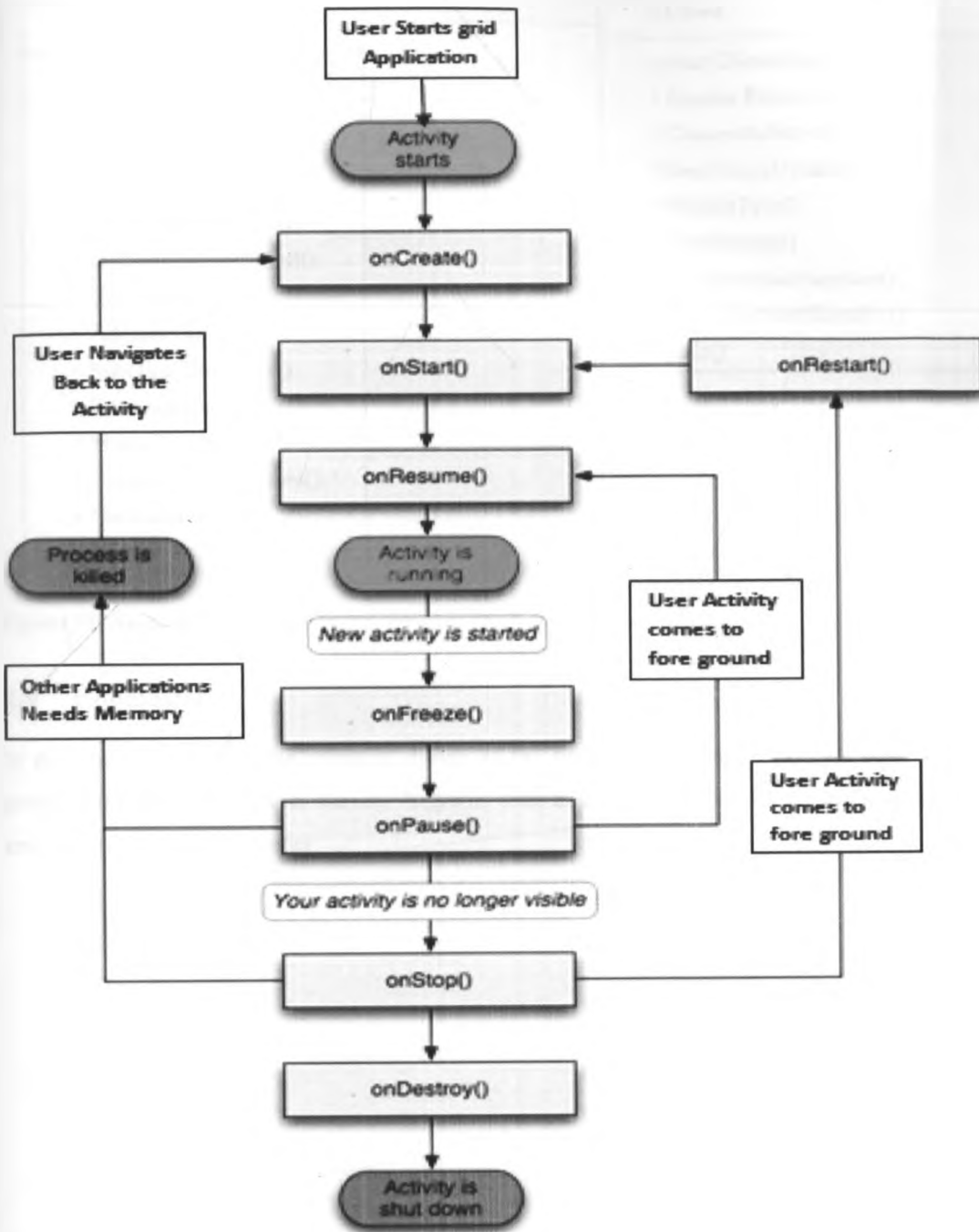


Figure 13: Activity Lifecycle Diagram

The below, shows components of in various modules used in the application and their respective properties and methods. The figure also shows the association between each class.

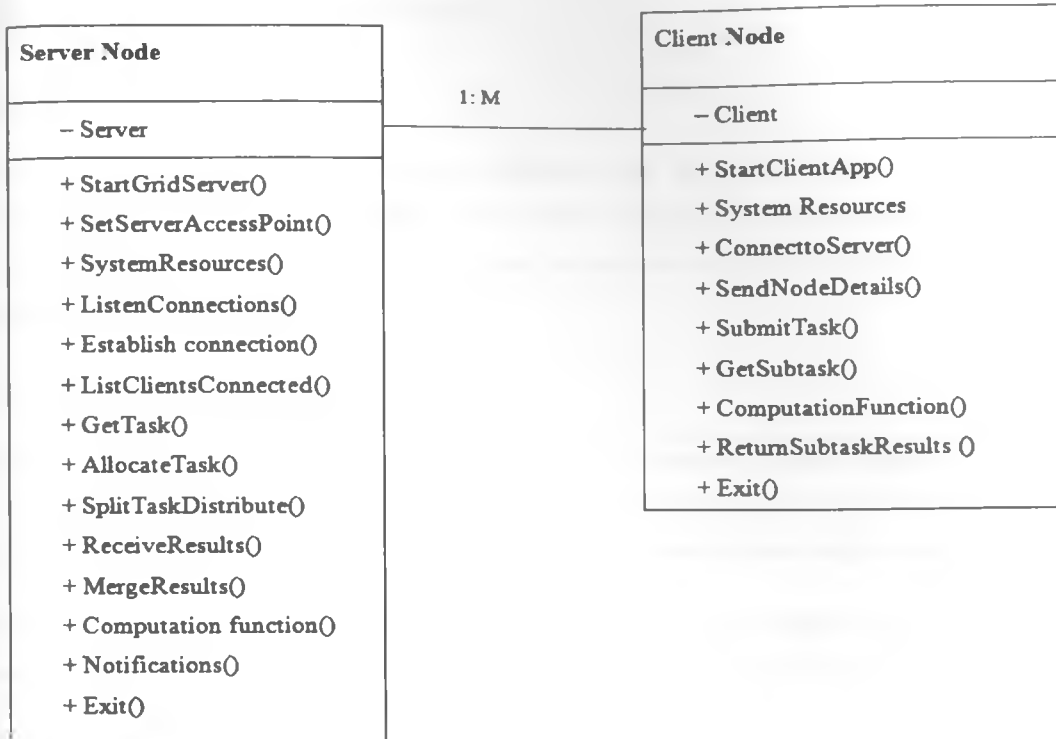


Figure 14: Association diagram

4.9 Conclusion

In this chapter presented the complete design of the system that included the core functional requirements and provided the general structure. Various diagrams were used to give detailed view of the components of the system and how they were put together.

CHAPTER 5: IMPLEMENTATION

5.0 Chapter Overview

This chapter describes in detail, how the middleware was implemented after overall design-issues had been addressed in the previous chapter. Focus in this chapter included, setting up development environment, how the various modules are implemented and how critical sections were secured. Screen shots and Code snippets are used along explaining important parts.

5.1 Implementation Environment

The implementation phase involved the translation of the designed system into an actual specific coded and executable software system. As the design of the system was based on the specified requirements discussed in the previous chapter, then the implementation should naturally account for these requirements.

The development of the implementation was coded using the Eclipse Integrated Development Environment (IDE) that provides a sufficient development platform for the underlying android SDK and Java Development Kit (JDK).

The environment setup involved installing the Android Software Development Kit (SDK) comes with a set of tools as well as a platform to run it and see it all work. Then installing the eclipse IDE and setting the tool access paths. Both android SDK and eclipse available as free downloads in android developer website, <http://developer.android.com> and eclipse website <http://www.eclipse.org> respectively.

The following steps were followed during the setup

- Step 1: Install the JDK
- Step 2: Install Eclipse IDE of Choice. In this project eclipse Helios is used
- Step 3: Install the Android SDK
- Step 4: Install the Android ADT for Eclipse
- Step 5: Create an Android Virtual Device (or AVD). The AVD is used to run emulators
- Configuring the android phone to run on USB debugging mode

5.2 Project Coding

Following successful project development environment, actual coding of the system started. The goal of the coding phase was to translate the design of the system into code in a given programming language. In this project java was used since it's the underlying language used in android SDK. In this phase consideration were observed to implement the design in the best possible manner. The coding phase affects both testing and maintenance profoundly. A well written code reduces the testing and maintenance effort. Hence, during coding the focus should be on developing programs that are easy to write. The system development of various modules was organized into packages and classes to enhance simplicity and clarity during the coding phase.

5.3 Permissions

In this project various permissions were required to access the operating system functionalities. A permission is generally a restriction limiting access to a part of the code or to data on the device. The limitation is imposed to protect critical data and code that could be misused to distort or damage the user experience. In the middleware development lot of access to this restricted system codes is necessary in order to provide abstraction to the user which is major requirement of the middleware.

The ability to start a particular Activity can be enforced when it is declared in its manifest's `<activity>` tag. By doing so, other applications will need to declare a corresponding `<uses-permission>` element in their own manifest to be able to start that activity.

The syntax is as follows: `<uses-permission android:name="string" />`

The following indicates some of the permissions used:

```
<uses-permission android:name="android.permission.INTERNET" />
```

```
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
```

```
<uses-permission android:name="android.permission.GET_TASKS" />
```

```
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
```

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

```
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

These permissions are defined in the manifest file components. The file is executed and it's used by the operating system to tell which components exist in, the application and additional information of the application.

5.4 Module Development

Following the methodology used, modular approach was used in the project for the middleware application development. The system was divided into various modules and implementation of each module done.

5.4.1 Main Graphical User Interface

The main graphical user interface was implemented using the tab activity classes provided as part of the android SDK environment. The tabbed interface was preferred due to ease of navigation, each tab representing main modules. This is implemented by the main grid package.

The screen shot below show the main user interface of the grid middleware



Figure 15: Main Graphical User interface of the server and client

5.4.2 System Resources Module.

The main purpose of this module was to create a layer of abstraction between the user and the android phone system by returning the status and values of various phone resources for instance battery level, memory location among others

5.4.3 Server Module:

In this module, a server system was implemented as the coordinator node which played the functions of managing other nodes communication, task allocation and result aggregation. It acts as a server, and listens for clients' connection on its IP address in a specific port. In this project port 8080 is used, the syntax is as follows < server IP>: 8080. The server node can act as a clients by participation in task computation, in this research project the server node splits the task as per the clients connected and its also assigns its self a part of sub divided task

5.4.4 Client Module:

In this module, client packages are implements user node: This refers to the client phone nodes that join the grid. This node takes a role of either a client when submitting a task or a worker when it is given some task to compute.

5.4.5 Interface Module:

This was implemented to log messages exchanged among the nodes communication with each other. The system was packaged into two applications, that is server and client application. This are compiled into .apk format before being installed on the android phone

5.5 Conclusion

To summarize, the middleware development was completed in a series of stages that involved different modules to achieve the complete system. Each stage presented a more refined set of challenges, from establishing the minimal functionality to providing the added feature aimed at achieving the projects ultimate goal and a functional middleware.

CHAPTER 6: TESTING AND RESULTS

6.0 Chapter Overview

This chapter documents the testing and results phase of the project. Testing is an extremely important stage of the development lifecycle of the middleware, ensuring that the system results satisfies all stated requirements and performs at a level that is satisfactory to the user.

6.1 Testing

6.1.1 Unit Testing

The project methodology used involved developing the project in modular manner. This resulted to “inspect-and-adapt” approach to development greatly reduces both development costs and time. Therefore unit testing helped to confirm that each component, or unit, of the system worked well by isolating the component as much as possible before testing. This isolation simplifies error detection when a problem arises. There are four main components in the system which have been subjected to unit testing – the system resource module, client, server and test application

6.1.2 Integration Testing

Integration testing was used to confirm that each component of the system integrated seamlessly with the others. The main modules were integrated to each other on the main GUI designed. The major work conducted at this stage includes verifying that the functions are performed accurately, that the system works with all interfacing modules systems, and that the middleware meets quality and/or standard requirements.

6.1.3 Performance Testing

The overall middleware performance testing involved testing that the system satisfied each non-functional requirement identified during the requirements analysis phase. The system’s performance was tested against these criteria:

- Compatibility with various version releases of android mobile OS
- Ability to for nodes to communicate
- Flexible messaging system, allowing the developer using the framework to determine what parameters to be passed when calling a service
- Performance – The middleware must load with ease and processing speed
- Ability warn the user if there is any error and to keep logs

The middleware system was tested, found to be running well and gave expected results.

6.1.4 Testing process

The tests were run on the android AVD emulator. Later when all aspects are confirmed working well, the test was done with the actual phone. This was achieved by setting the phone on debug mode to ensure that one can run and install the application directly from the laptop while the phone is connected. Tests were performed against the functional and non-function requirements discussed in previous chapters. For instance, the screen shot below show the application being tested on the AVD emulator and returning results before it's actually tested on the phone.



Figure 16 (a) Grid Server tested on emulator

Figure 16 (b) Grid Server tested on phone

Figure 16: Module testing diagram on emulator then on phone

The figure (a) above show the server module being tested on the emulator and its listening for the clients on IP 10.0.2.15 which is the emulator default virtual IP address. Then after it's confirmed working it's compiled and tested on the actual phone, the figure (b) shows the module tested on the actual phone and it's listening for clients on a public IP over 3G mobile provider network.

6.1.5 Emulator Setup and Testing

The application was fully tested using emulators. To test the application on the emulators, we had to configure multiple Android emulators as server and clients to run on the same computer. In this case, two emulators a server and two clients were configured. The server emulator run on emulator 5554 and the clients run on emulator 5556. However in android, the emulators cannot communicate directly since each emulator is in a virtual network, with a virtual router and all emulators has the same ip address. This coupled with inability of emulators to support WIFI formed the major limitation of using emulators to simulate many clients in a grid environment. In this regard, port redirection was used by using telnet between the emulators as shown in the code snippet below, where 8080 is server port on which it's listening for connection and client port is 8081.

- i. Telnet local host 5554
- ii. Redir add tcp:8081:8080

The diagram below show the screen short of telnet emulator redirection setup and output of the client and server emulators with the final result achieved after successful testing

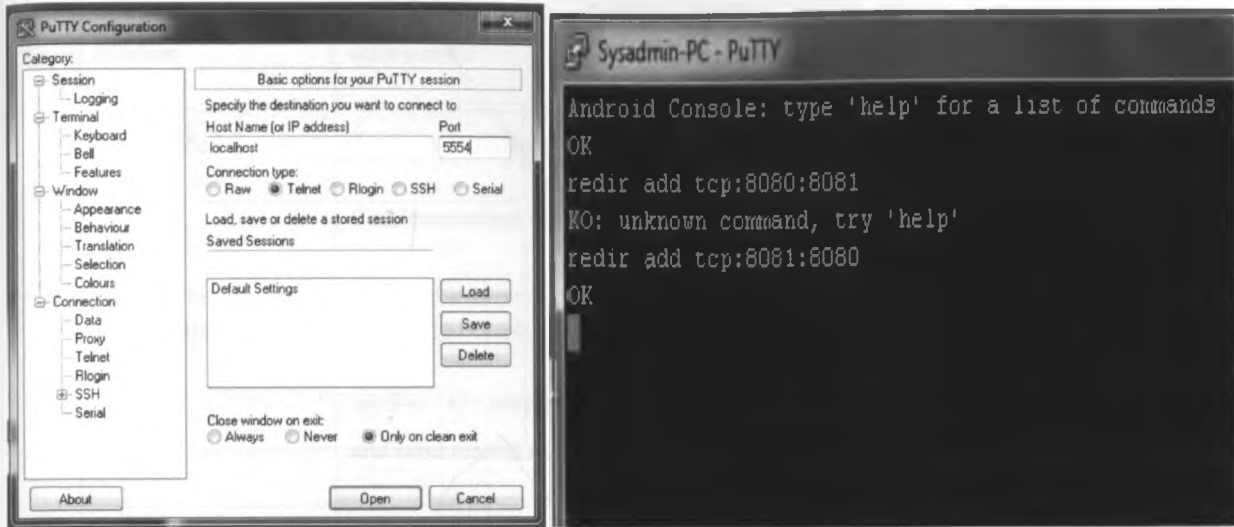


Figure 17: Emulator telnet and port redirection setup



Figure 18: Server and client emulator displaying results

6.1.6 Testing and Evaluation Matrix

The tables below show a summary matrix of some testing done, indicating that all tests were successful.

No.	Module	Test Case	Actual Result	Pass/Fail
1	System Resources Module	Test ability to get system information	Returns System battery level on real time. Other properties returned well	Pass
2	Server Module	Server initialized on Communication with client	Server initialized well, it reports its own IP on which it is listening for the clients. Successful client connection	Pass
3	Client Module	Ability to connect to server Ability to compute work and send results to server	Client connects successfully Client does the work assigned and submits back results	Pass
4	Application	The application involves computing work assigned by server and client doing the work	Server and clients interacts well and performs the work successfully	Pass
5	Integrated GUI with all modules	Test navigation	Ability to navigate to all tabs specified	Pass
6	Emulator communication	Client and server application communication via emulator	Telnet and port redirection to allow emulator interconnection. Connection established and data sent across	Pass
7	Multi Phone Testing (Actual Mobile Phone Grid Testing)	Applications tested with two clients and one server/ coordinator node	Test successful, clients connected to server and exchanged data	Pass

Table 3: Testing summary

Fully tested modules were integrated, tested and finally the whole middleware was tested on emulators before tested on the android phones. System testing was done on completely integrated system to verify that it meets its requirements. The system was evaluated by having it tested by fellow students using actual phones. The middleware run well and different phones and communication and data exchange was successful demonstrating its applicability in real world environment.

6.2 Results

The compiled applications were installed on the android phones as clients or the server. The network was setup by configuring android phone on access point. The phones are connected to the network, assuming its WIFI based local network. Alternatively internet access was enabled on the phones to establish a network, incase WIFI access was not available. Then the server was initialized by clicking start button. The server returns its IP address, which is keyed on the client. On clicking “Connect” on the client, the server accepts the incoming socket connection request. Successful connection is established and then the work to be done is submitted for computation.

The following results were successfully achieved.

- i. Development of the both client and server node applications
- ii. Successful communication using sockets in a WIFI network. Ability of client to submit a task request, to the coordinator
- iii. Ability of the coordinator generate computational load, split the task and to submit subtask work to client for computation and results returned back
- iv. Ability of the client node to send request, receive and compute subtask from coordinator and sending the subtask result to coordinator
- v. Ability to read system resources
- vi. Successful testing the node on emulator and finally using phones in real work environment. The communication was successful over mobile operator network where WIFI was not available, proving viability of grid middleware

6.3 Screen shots of the results and description

6.3.1 System Resources Results

The screen shot show the node system resource information’s information. These resources are used in grid to tell the resource status of a node for instance if the node has low battery level.



Figure 19: System Resource diagram

6.3.2 Server Node Operation

The diagram show the server being initialized, listening on a particular IP and port and finally when it's connected to the client



Figure 20(a) Server Node



Figure 20(b) Server Node initialized

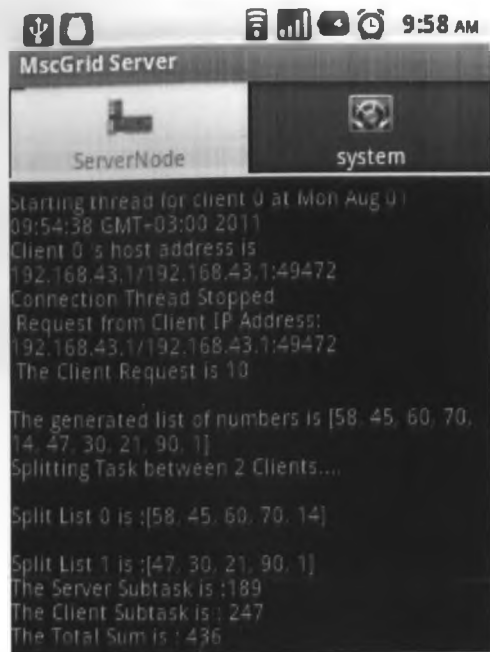


Figure 20(c) Server Node displaying logs

Figure 20(d) Server Node show operation logs

Figure 20: Server node operation diagrams

The diagram shows the process of the server being initialized by clicking start button, then the server updates its status giving the time started and the IP it's listening for incoming clients on.

Once the client successfully connects to the server, the server starts a new thread for the client. The server receives a task request from client defining the task load size.

In this case it receives request to generate 10 random numbers. The server generates the numbers, splits the task to connected clients and distributes the task to compute to the connected clients for computation by splitting the task. The server receives individual results from clients and sums them to get final results

6.3.3 Client node Operation

The Screen shot on figure 22 illustrates the client node operation. This includes client initialization, connection to server, submission of a job request, assignment of a subtask by the server, clients doing the task and receiving final result computed by server.

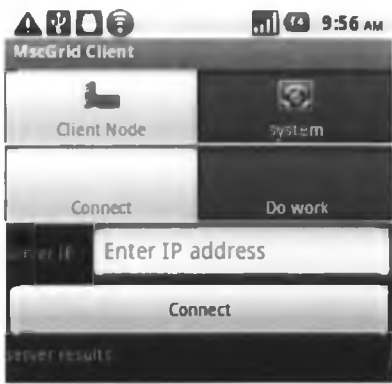


Figure 21 (a). Client before Connecting

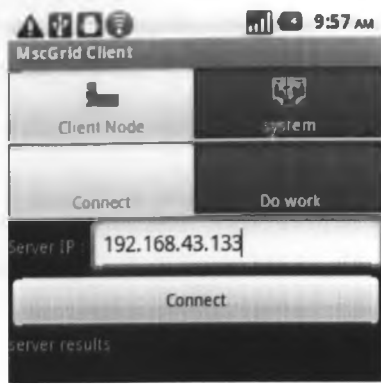


Figure 21 (b). Client with IP of Server Entered

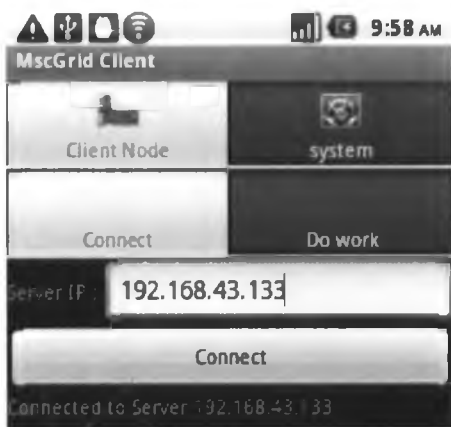


Figure 21 (c). Client successfully connected to server.

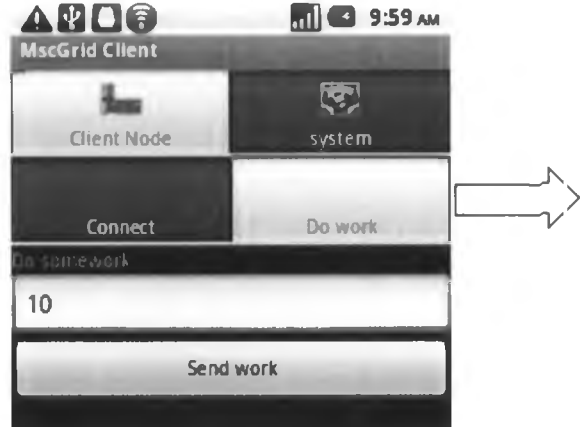


Figure 21 (d). Client Sending work request load size 10



Figure 21 (e). The client receive sub task assigned by server, computes and returns subtask to server node for compiling to final result. Then the client (Requesting node) receives final results from server

Figure 21: Server node operation diagrams

The Client node Operation diagrams in shows the process of how the clients is setup to connects to server, notifications of successful connection, sending work to client and finally displaying received results.

6.4 Conclusion

The tests were successfully run different modules to achieving the goal of the complete system. The middleware communication and task handling was achieved acting as a proof of concept that grid computing is possible on mobile phones.

CHAPTER 7: DISCUSSION

7.0 Chapter Overview

This chapter presents achievements, challenges, limitations, brief description of the potential future works and conclusion of this work done in this project. The chapter indicates the necessity for more future research in this domain so as to enable the implementation of the fully functional middleware in Mobile Grid environment.

7.1 Achievements

The project achieved its goals, fulfilled user requirements and functioned adequately in the role that was intended. The middleware was successfully tested using emulators and actual android phones.

The following results were successfully achieved.

- i. Development of the both client and server node applications
- ii. Successful communication using TCP sockets in a WIFI network and over mobile Telephony provider GSM or 3 G network
- iii. Ability of client to submit a task request, to the coordinator
- iv. Ability of the coordinator generate computational load,
- v. The server ability to split the task and to submit subtask work to client for computation and results returned back
- vi. Ability of the client node to send request, receive and compute subtask from coordinator and sending the subtask result to coordinator
- vii. Ability to read system resources
- viii. Successful testing both nodes that is server and client on emulator and finally using two Ideos android phones in real work environment. The communication was successful over mobile operator network where WIFI was not available, proving viability of grid middleware. The application was also successfully tested on Samsung Galaxy Tab which established communication with Ideos phone as a proof of different manufacturer's interoperability.

7.2 Challenges

Grid computing on mobile phones is relatively new and few research materials were available for reference. This meant that extensive research was done online.

Though smart phones are the most promising heralds of future pervasive computing, developing middleware applications that can properly perform in this new computing landscape is not easy.

Middleware development usually is usually a broad task with various aspects that that needs to be addressed, thus middleware platforms can be complex, for example due to complexity of network communication, fault tolerance, and security among other components

The project turned out to be more challenging than had been anticipated. Since android mobile operating system is relatively new, having been launched on year 2007, no much work had been done previously on mobile grid computing based on android, a lot of original solutions had to be created.

Though, android mobile operating system is open source, most smart phones are rooted, meaning there is no super user (SU) privileges. Access the root of the phone required hacking of the operating. This has implication of having the phone out of warranty due to breaking the OS to access the root. In this regard, adhoc communication could not be achieved.

There was a challenge in coming up with methods for establishing communications between the devices. What protocols should be used and how to enforce integrity of transmitted data?

Implementing middleware platform for mobile environments has to cope with a wide diversity of challenges due to limitations like; mobility, heterogeneity, limited resources and irregular connectivity.

The methodology used required testing each requirement in iterative manner thus development methods do not scale. Due to the integrative approach, it is hard for some to understand exactly where the project stands.

7.3 Limitations of the System.

The android mobile operating system does not support adhoc communication through WIFI, in order to use WIFI in adhoc manner; it required hacking the operating system.

Security: Due to the scope of the project, not much attention was paid to enforcing security. Messages are passed as unencrypted socket objects and their source is not authenticated.

Inability to save system state in case of an emergency failure. This requires check pointing mechanisms to be implemented on the middleware in future. Thus need for check pointing.

Inability to discover nodes automatically, Node discovery will help the server node to automatically discover all the clients on the same subnet. Currently the node for the nodes to connect to the server node one has to manually key in server IP and the then connect.

Clock Synchronization. In grid computing communication needs to be synchronized, thus clock synchronization services would need to be provided for applications that rely on accurate timing for effective operation.

Android emulators run on virtual networks that have virtual router, thus each emulators have same ip address. This posed a challenge during testing using emulators since it required use of port redirection via telnet. This coupled by the fact that emulators don't support WIFI as the system was designed was a challenge that necessitated used of actual mobile phone and purchasing of two android based smart phones.

7.4 Recommendations for Further Work

This mobile grid technology provides endless possibilities for developing the middleware further with the option integrating various features for the benefit of the target audience. To enhance this grid middleware prototype and to increase the usability of the overall solution the following items may be considered for future implementation.

Incorporate more system resources parameters in finding the system node for instance use of CPU, Memory and Network signal strength in computing a node or client system strength score in a grid

Incorporate maps: the system resources module has capability of getting the location of the node. Therefore a map layer can be included on the middleware to help to indentify actual locations of the participating nodes, since they can communicate via the mobile provide network. However both client and server should have a subset of map application in order to limit amount of data transmitted to conserve bandwidth.

Check pointing: This is the process of periodically saving intermediated data and machine states on reliable storage during the course of a long running application, so that in the event of a failure, the application can be recovered from the checkpoint saved prior to the failure, instead of starting the application from the beginning. In mobile grid computing systems, hosts are interconnected wirelessly and move at will. As a result, check pointing becomes more complicated in a mobile grid computing system than in its conventional distributed computing counterpart, where hosts and nodes are interconnected.

Security: the communication between the middleware handles crucial information and it's transmitted over WIFI network thus under various security threats. There is need to implement a form of two way encryption ensure ensuring message confidentiality and integrity. However, the use of session keys is a possible solution, preferably a different session key for every pair of communicating nodes. A way of authenticating messages, say through use of sequence numbers and discarding messages not coming from a known set of IP addresses and port numbers, may also be used to improve security.

Node Discovery: Implement and test the discovery protocol between nodes of collaborating phones in ad-hoc mode. This will involve implementing ad hoc communication over WIFI, not supported by android operating system and collaborating with another group and installing the same software on both phones.

The growing success of mobile computing devices and networking technologies, such as call for the investigation of new middleware that deal with mobile computing requirements, in particular with context-awareness, resource awareness and addressing various mobile phone limitations like mobility and heterogeneity

The android mobile operating system should be enhanced to support the following; adhoc communication through WIFI, android Emulators to support WIFI, Emulator to have different network address to allow emulator communication without port redirection via telnet and allow phone root access.

Finally, In future middleware implementation should continue to focus on handling security, improving support for offline processing and presentation of results depending upon the device. Along with this implementation intend to continue validating the approach by experimental results enhance the functionality of the middleware in future version releases.

7.5 Conclusion

The research project was a proof of concept that grid computing can be achieved on mobile phones by utilizing a middleware, while exploring the android operating which is open source based mobile operating system framework. The results achieved were demonstration of the concept using the middleware, developed and tested using various mobile phones as server and client's nodes. An application was tested on the middleware, successfully demonstrated resource sharing, mobility and ability to collaborate in task computation. The middleware was successfully tested with two android phones and results achieved fully indicating potential uses of smart mobile phones in grid environment. The middleware for mobile grid computing on android enhanced collaborating collection of mobile devices to optimally use available resources. It provided a layer of abstraction that enabled mobile phones on a grid act as collaborating nodes, extending grid computing to mobile devices achieving mobility and resource sharing. The research project achieved its purpose to extend distributed computing to mobile phones utilizing android mobile operating system platform.

REFERENCES.

Advanced Technology Explored [online]. (2010). Available from: <http://www.aeili.com/advantages-and-disadvantages-of-grid-computing/> [Accessed 23 Feb 2011].

Android 2011[online]. (2011). Available from: <http://www.android.com/about/> [Accessed 12 March 2011].

Android Developers [online]. (2011). Available from: [http:// developer.android.com /guide/basics/what-is-android.html](http://developer.android.com/guide/basics/what-is-android.html). [Accessed 18 March 2011].

Antonios L, Dimitrios S and Theodora V. (2007). Mobile Grid Computing: Changes and Challenges of Resource Management in a Mobile Grid Environment, National Technical University, Athens.

Bart, J., Brown, K. and Fukui, N. (2005). Introduction to Grid Computing. Texas: International Business Machines Corporation. pp3-10. Katarina S, Thomas W and Santi Ristol.(2010).

Britni M, Kathleen L and John E (2010). Mobile Money: Cell Phone Banking In Developing Countries. Policy matters Journal. 7, p.27-34.

Capra L, Emmerich W, Mascolo C. (2002). Middleware for Mobile Computing: Department of Computer Science University College London

David, C and Marty, H. (2004). Mobile OGS.NET: Grid Computing on Mobile Devices, ACM International Workshop on Grid Computing, Pittsburgh, PA.

Folding@home [online]. (2010). Available from: <http://folding.stanford.edu/English/Main>. [Accessed 22 Feb 2011].

Foster I. (2002) "What is the Grid? A Three Point Checklist", GRID Today

Gail B. (2010). Agile Methodology – Delivering Projects Faster, Cheaper, Better. CIO Practice, Taos

Jung D, Paek K, and Kim T. (1999). Design Of MOBILE MOM: Message Oriented Middleware Service for Mobile Computing Department of Computer Science & Engineering, Korea University

Junseok, H and Praveen, A. (2004). Middleware Services for P2P Computing in Wireless Grid Networks, IEEE Internet Computing

Katarina S, Thomas W and Santi R. (2010). Grid and Cloud Computing, A Business Perspective on Technology and Applications. St. Gallen, Switzerland: Springer. pp 23-125.

KNBS. (2010). Census Highlight. ? [online] Available from: <http://www.knbs.or.ke/Census%20Results/KNBS%20Brochure.pdf>. [Last accessed 24th Feb 2011]

Licia, C, Cecilia, M., Stefanos, Z and Wolfgang, E. (2001). Towards a Mobile Computing Middleware: A Synergy of Reflection and Mobile Code Techniques Department of Computer Science, University College London, Gower Street, London

Manvi, S and Birje, M. (2010). A Review on Wireless Grid Computing,. International Journal of Computer and Electrical Engineering. Vol. 2, No. 3, pp.469-474

Masinde, M., Antoine B. And Victor M. (2010). Middleware for Grid Computing on Mobile Phones. In: Enrique, C. and Marco, Z. m-Science, Sensing, Computing and Dissemination. Italy: ICTP. pp 169-188.

Masinde, M., Antoine, B. And Victor, M. 2010. MobiGrid: A Middleware for Integrating Mobile Phone and Grid Computing. In The 6th International Conference on Network and Service Management (CNSM 2010), Niagara Falls, Canada, October 25 - 29, IEEE Communications Society.

Muruganatham S, Srivastha P and Khanaa (2010). Object Based Middleware for Grid Computing. Journal of Computer Science. 6, pp.336-340.

Ndegwa, V. (2009). Distributed Mobile Application Framework , University of Nairobi

Oriana, R. (2008). Challenges and Lessons in Developing Middleware on Smart Phones. In: J. Kangasharju Computing Practices. Zurich: IEEE Computer Society. 77-85.

Seti@Home [online]. (2010). Available from: <http://setiathome.ssl.berkeley.edu/> [Accessed 22 Feb 2011].

Umar, K, Hassan, J, Ali, S and Sungyoung, L. (2005). Mobile-to-Grid Middleware: An approach for breaching the divide between mobile and Grid environments. Department of Computer Engineering, Kyung Hee University

Wireless Federation [online]. (2011). Available from: <http://wirelessfederation.com/news/11902-kenyas-mobile-market-kenya/>. [Accessed 10 Jan 2011].

APPENDICES

Appendix A: User Guide

Middleware for Mobile Phone Grid based on Android User Guide

Introduction

Android Grid middleware application is a mobile grid computing middleware based on android mobile operating systems which is open source.

The middleware technologies provide abstraction platform built on top of network operating systems, to enhance the design and implementation of distributed applications.

It allows various nodes to communicate with any other node having the same software running and connected to the same network platform. Users of various nodes can collaborate taking advantages of abstraction layer introduced by the middleware to participate in Grid computing. In this regard the application application helps to harness the raw computing power of mobile phones platform to addressing mobility and resource sharing in mobile grid environment which is not fully utilised most of times

Prerequisites

The project I required a diverse of resources; (this can be categorized as the following.

Equipments in case you are running it on a computer as eclipse project which is not compiled as .apk:)

- i. Laptop: Running of windows 7 home premium as the development machine
- ii. Two android powered smart phones (Huawei Ideos running android 2.2 proposed for the project)

Software:

- i. Java SDK 2.2 with the latest (0.97)
- ii. Android Virtual Device (AVD) plug-in for Eclipse
- iii. Eclipse IDE for Java Developers Version: Helios Service Release 1
- iv. Adroid Grid middleware eclipse project (Not Compiled)
- v. Adroid Grid middleware .apk package (Compiled)

Installing the Application

a) Installing directly from the computer

In this case the android middleware application is not compiled as APK file

Connect phone to the laptop running the android SDK and Eclipse project

Ensure USB Debug mode is enabled under settings ---> applications ---> Development --->USB Debugging

Click on the android grid Middleware project, then run the project and select the android mobile device

The installation is automatically done; the progress can be seen on the eclipse console

The application is automatically launched

b) Installing Compiled APK file

Before performing these steps, be sure to enable your Droid to allow installation from third party sources.

Connect the device to the computer the device should show up as a drive.

Ensure you enable “Allow installation from Unknown sources ” under application settings. This is because the application is not signed

Copy the (android grid middleware) APK file to the root directory of your memory card.

Open the **ApkInstaller** app or browse to where the APK file is copied

It will show the files on your SD Card. Tap the APK file to run the installer.

Tap **Install**.

On successfully installation the application is automatically launched or can be accessed on the phone and the launched.

Android Phones setup

In order to be able to run and use the grid middleware applications, a few configuration steps are required.

Setting up WIFI access point: Click on Settings ---> Wireless & Networks ---> Enable WIFI

The setup the Hotspot: Click on Settings ---> Wireless & Networks ---> Tethering and Portable hotspot ---> enable Portable WIFI Hotspot

Applications

The project comprises of two Applications

- i. Server application
- ii. Client Node Application

The client node application is a subset of the server application without many classes. However it has a class that enables it compute the task given by the server node

Application Nodes:

Server node: acts as the coordinator node which plays the functions of managing other nodes communication, task allocation and result aggregation. It acts as a server, and listens for clients connection on its IP address in a specific port. In this project port 8080 is used.

Client Node: This implements user node, which refers to the client phone nodes that join the grid. This node takes a role of either a client when submitting a task or a worker when it is given some task to compute.

Initializing the Server

Install the server application.

The application is automatically launched. This can be also launched from the application icon

The screen shot show sever application launched

Click start to initialize

The server indicates its listening for clients connections on server IP on the defined port

Setting the Client phones

Install the client application.

The application is automatically launched. This can be also launched from the application icon

The screen shot show sever application launched

Enter the Server IP address

Click Connect

The client reports successful connection

Integrating new application on the middleware

The source code module of the new application can be introduced on the middle to extend the functionality. This is achieved by replacing the computing module on the middleware in server application and modifying .the client node application. This ensure scalability of the grid application and support of the functionalities in the grid

Appendix C: Sample Source Code

The source code, copy of the application and documentation is available on the project CD submitted