PUNE RESEARCH SCHOLAR ISSN 2455-314X
AN INTERNATIONAL MULTIDISCIPLINARY JOURNAL VOL 3, ISSUE 2

A Computation Offloading Framework for Efficient Energy Usage in Smart Mobile Devices through Mobile Cloud Computing

ARTHI. G[1], JAYASHREE R[2], SOWMYA. M[3]

Department of Information Technology
Easwari Engineering College

Abstract— **The widespread use and increasing capabilities of mobile devices are making them a viable platform for offering mobile services. However the increasing resource demands of mobile services and the inherent constraints of mobile devices limit the quality and type of functionality that can be offered, preventing mobile devices from exploiting their full potential as reliable service providers. Computation offloading offers mobile devices the opportunity to transfer resource–intensive computation to more resourceful computing infrastructures. In this paper we present a framework for cloud assisted mobile service provisioning to assist mobile devices in delivering reliable services. It also enables the mobile provider to delegate the cloud infrastructure to forward the service response directly to the user when no further processing is required by the provider.**

I. Introduction

In the last decade we have seen, and continue to see, a wide adoption of advanced mobile phones, called smart phones.
These smart phones typically have a rich set of sensors and radios, a relatively powerful mobile processor as well as a substantial amount of internal and external memory. A wide variety of operating systems have been developed to manage these resources, allowing programmers to build custom applications. Centralized market places, like the Apple App Store and the Android Market, have eased the publishing of applications. Hence, the number of applications has exploded over the last several years – much like the number of web pages did during the early days of the World Wide Web – and has resulted in a wide variety of applications, ranging from advanced 3D games, to social networking integration applications, navigation applications, health applications and many more. Not only has the number of third-party applications available for these mobile platforms grown rapidly – from 500 to 185,000+ applications within two years for the Apple App Store –, but also the smart phones' processor speed increased along with its memory size, the screen resolution and the quality of the available sensors. Today's smart phones offer users more applications, more

communication bandwidth and more processing, which together put an increasingly heavier burden on its energy usage, while advances in battery capacity do not keep up with the requirements of the modern user. Recently, it has been rediscovered that offloading computation using the available communication channels to remote cloud resources can help to reduce the pressure on the energy usage.
Furthermore, offloading computation can result in significant speedups of the computation, since remote resources have much more compute power than smart phones.

In this paper we present a framework for computation offloading. The framework is targeted at the Android platform, since Android provides an application model that fits well for computation offloading. The framework offers a very simple programming model that is prepared for mobile environments, such as those where connectivity with remote resources suddenly disappears. It supports local and remote execution and it bundles both local and remote code in a single package.

The primary objective of the proposed system is to develop a framework to

- ➢ Perform cloud assisted mobile service provisioning to extend the capabilities of smart mobile devices and to store the results of the computation in a cloud so that it can be retrieved whenever needed.
- ➢ Perform dynamic offloading of user task so that resource intensive computations can be performed in a more resourceful infrastructure.
- ➢ Provide the facility to share the results of the computation over the social media by connecting to the applications available in the user mobile.

II. BACKGROUND

The framework proposed in this paper is based on the concept of mobile cloud computing. The framework is develop as an android application that also uses the open source framework apache tomcat, J2EE Technologies. Android is an open source platform including an operating System, middleware and key applications and is targeted at smart phones and other devices with limited resources. Android has been developed by the Open Handset Alliance, in which Google is one of the key participants. Android applications are written in Java and then compiled to Dalvik byte code and run on the Dalvik Virtual Machine.

A. Android Application Components

The main components of Android applications can be categorized into Activities, Services, Content Providers, and Broadcast Receivers, which all have their own specific lifecycle within the system. Activities are components that interact with the user, they contain the user interface and do basic computing. Services should be used for CPU or network intensive operations and will run in the background; they do not have graphical user interfaces. Content Providers are used for data access and data sharing among applications. Finally, Broadcast Receivers are small applications that are triggered by events which are broadcasted by the other components in the system.
For computation offloading, we focus on activities and services, because the separation between the large computational tasks in the services and the user interface tasks in the activities form a natural basis for the Cuckoo framework. We will now have a closer look at how activities and services communicate in Android.

B. Android IPC

When a user launches an application on a device running the Android operating system, it starts an activity. This activity presents a graphical user interface to the user, and is able to bind to services. It can bind to running services or start a new service. Services can be shared between multiple activities. Once the activity is bound to the running service, it will communicate with the service through inter process Communication, using a predefined interface by the programmer and a stub/proxy pair generated by the Android compiler. Service interfaces are defined in an interface definition language called AIDL. Service methods are invoked by calling the proxy methods. The Android IPC also supports call backs, so that the service can invoke a method on the activity, allowing for asynchronous interfaces between activities and services.

C. Android Application Development

Android applications have to be written in the Java language and can be written in any editor. However, the recommended and most used development environment for Android applications is Eclipse [7], for which an Android specific plugin is available [2].Eclipse provides a rich development environment, which includes syntax highlighting, code completion, a graphical user interface, a debugging environment and much more convenient functionality for application developers. The build process of an Android application will be automatically triggered after each change in the code, or explicitly by the developer. The build process will invoke the following builders in order:

• Android Resource Manager generates a Java file to ease the access of resources, such as images, sounds and layout definitions in code.
• Android Pre Compiler generates Java files from AIDL files
• Java Builder compiles the Java source code and the generated Java code
• Package Builder bundles the resources, the compiled
code and the application manifest into a single file After a successful build, an Android package file (.apk) is created.

D. Computation Offloading

A basic computation-offloading system is composed of a client component running on the mobile device and a server component running in the cloud. The client component has three major functions. First, it monitors and predicts the network performance of the mobile device. Second, it tracks and predicts the execution requirements of mobile applications in terms of input/output data requirements and execution time on both the mobile device and the cloud. Third, using this information the client component chooses some portions of the computation to execute in the cloud so that the total execution time is minimized. The server component executes these offloaded portions immediately after receiving them and returns the results back to the client component so that the application can be resumed on the mobile device.

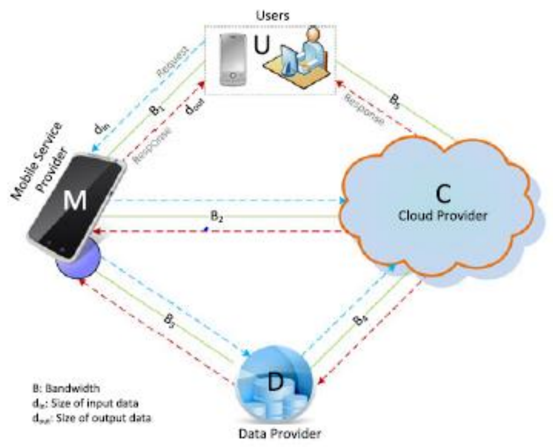


Fig. 1. An abstract view of cloud-assisted mobile service architecture,

Showing possible interacting entities and context information, where B is the link bandwidth and d_{in} and d_{out} represent the amount of data exchange over a link in both directions.

Computation offloading trades off communication cost for computation gain. Previous systems usually assume stable network connectivity and adequate cloud computation resources. However, in mobile environments a mobile device may experience varying or even intermittent connectivity, while cloud resources may be temporarily unavailable or occupied. Thus, the communication cost may be higher, while the computation gain will be lower. Moreover, the network and execution prediction may be inaccurate, causing the performance of these systems to be degraded. Cloud Computation Resources Cloud computation resources are usually provided in the form of virtual machine (VM) instances. To use a VM instance, a user installs an OS on the VM and starts it up, both incurring delay. VM instances are leased based on a time quanta. e.g., Amazon EC2 uses a one hour lease granularity. If a VM instance is used for less than the time quanta, the user must still pay for usage. A cloud provider typically provides various types of VM instances with different properties and prices. This server component needs to be launched at the time the offloading request is made and terminated when the required computation is complete.

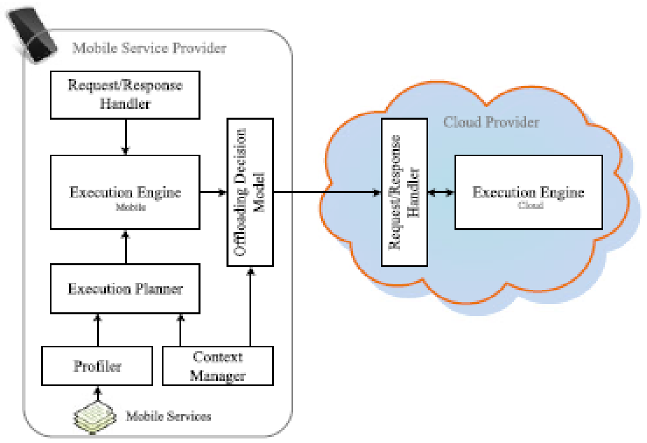


Fig. 2. The architecture of the cloud-assisted mobile service provisioning.

The lifetime of the server component is typically much less than the lease quantum used by the cloud service provider. An important question we consider in our system design is how to ensure there is enough VM capacity available to handle the mobile computation load without needing to always launch VM instances on-demand and incur long setup time.

E. Mobile Cloud Computing:

Mobile Cloud Computing (MCC) is the combination of cloud computing, mobile computing and wireless networks to bring rich computational resources to mobile users, network operators, as well as cloud computing providers. The ultimate goal of MCC is to enable execution of rich mobile applications on a plethora of mobile devices, with a rich user experience. MCC provides business opportunities for mobile network operators as well as cloud providers. More comprehensively, MCC can be defined as a rich mobile computing technology that leverages unified elastic resources of varied clouds and network technologies toward unrestricted functionality, storage, and mobility to serve a multitude of mobile devices anywhere, anytime through the channel of Ethernet or Internet regardless of heterogeneous environments and platforms based on the pay-as-you-use principle. MCC uses computational augmentation approaches (computations are executed remotely instead of on the device) by which resource-constraint mobile devices can utilize computational resources of varied cloud-based resources.[1] In MCC, there are four types of cloud-based resources, namely distant immobile clouds, proximate immobile computing entities, proximate mobile computing entities, and hybrid (combination of the other three model). Giant clouds such as Amazon EC2 are in the distant immobile groups whereas cloudlet or surrogates are member of proximate immobile computing entities. Smart phones, tablets, handheld devices, and wearable computing devices are part of the third group of cloud-based resources which is proximate mobile computing entities.

F. Apache Tomcat

Apache Tomcat, often referred to as Tomcat Server, is an open-source Java Servlet Container developed by the Apache Software Foundation (ASF). Tomcat implements several Java EE specifications including Java Servlet, JavaServer Pages (JSP), Java EL, and WebSocket, and provides a "pure Java" HTTP web server environment in which Java code can run. Tomcat is developed and maintained by an open community of developers under the auspices of the Apache Software Foundation, released under the Apache License 2.0 license, and is open-source software.Apache Tomcat version 7.0 implements the Servlet 3.0 and JavaServer Pages 2.2 specifications from the Java Community Process, and includes many additional features that make it a useful platform for developing and deploying web applications and web services.

G. J2EE

J2EE is a platform-independent, Java-centric environment from Sun for developing, building and deploying Web-based enterprise applications online. The J2EE platform consists of a set of services, APIs, and protocols that provide the functionality for developing multitiered, Web-based applications.

Some of the key features and services of J2EE:

- At the client tier, J2EE supports pure HTML, as well as Java applets or applications. It relies on Java Server Pages and servlet code to create HTML or other formatted data for the client.
- Enterprise JavaBeans (EJBs) provide another layer where the platform's logic is stored. An EJB server provides functions such as threading, concurrency, security and memory management. These services are transparent to the author.
- Java Database Connectivity (JDBC), which is the Java equivalent to ODBC, is the standard interface for Java databases.
- The Java servlet API enhances consistency for developers without requiring a graphical user interface

III. IMPLEMENTATION

A. System Modules

Our proposed system presents a distributed mobile service provisioning framework that reduces the burden on mobile resources through the offloading of resource intensive processes to the cloud. An offloading decision model is proposed to determine whether or not remote execution of a resource request brings performance improvements. The decision making involves selecting the best available resource provider according to the resource availability. This framework dynamically allocates resource providers to offload tasks to best suit the task requirements and environment context. The decision maker determines the best execution plan that achieves highest performance gain. The proposed system can be split into four modules.

- ***User Registration and Task Delegation***

In this module the user first registers himself before assigning the task to the service provider. After registration the user delegates the task to the service provider.

- ***Offloading Decision Making***

In this module the service provider selects the node to which the task has to be offloaded. The decision making involves selecting the best available resource provider according to the resource availability and performance measures. The decision maker determines the best execution plan that achieves the highest performance gain.

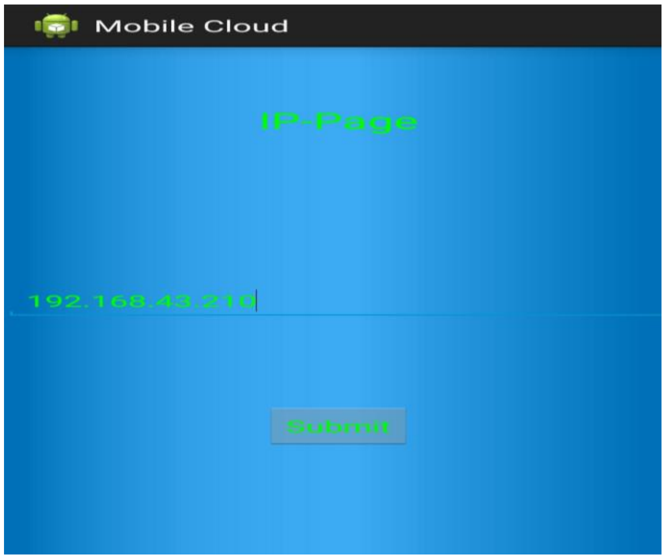
- ***Output Retrieval and Cloud Storage***

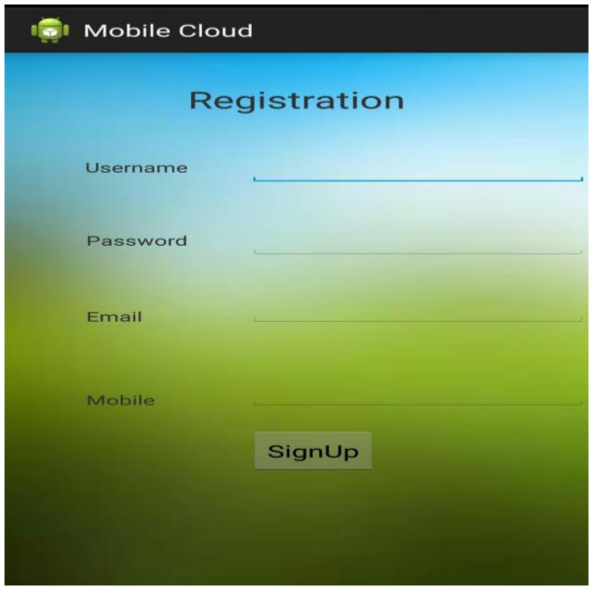
In this module the results of the computation are sent back to the user mobile directly after the task is completed by the offload node and a copy of the result is stored in the cloud for future usage.

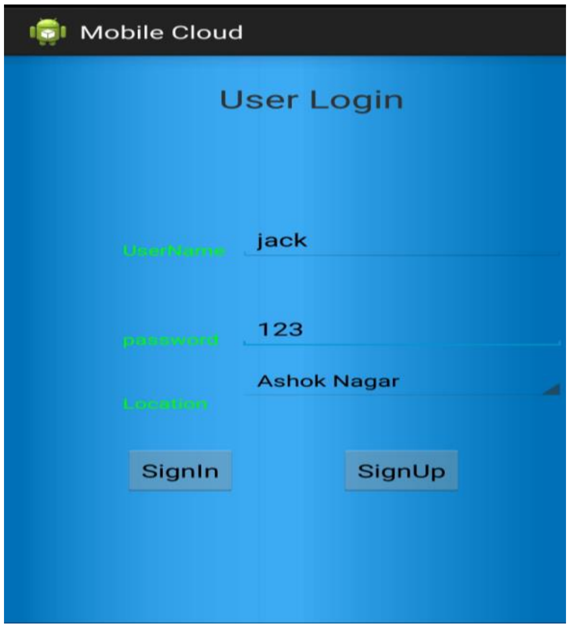
- ***Social Media Sharing***

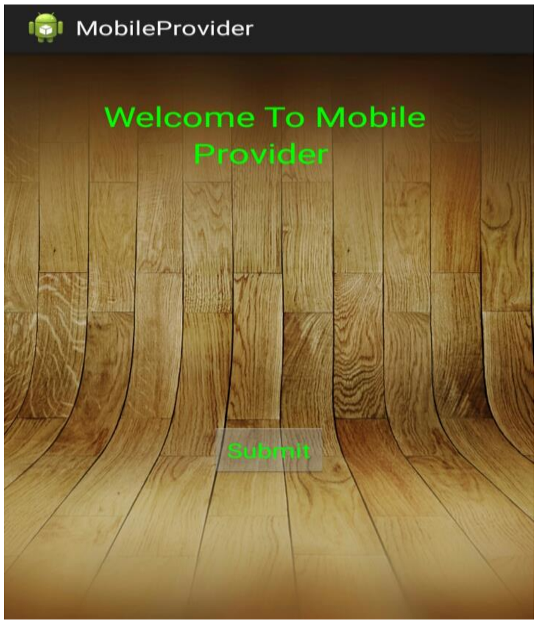
This module provides the facility to the user to share the results of the computation over the social media using an application of his/her choice. This module lists the available applications on the user mobile and allows the user to select one among them to share the results.

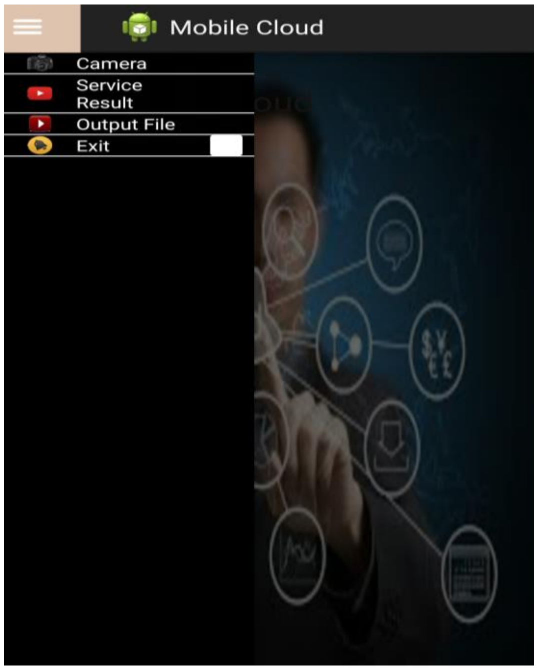
B. Implementation Snapshot:

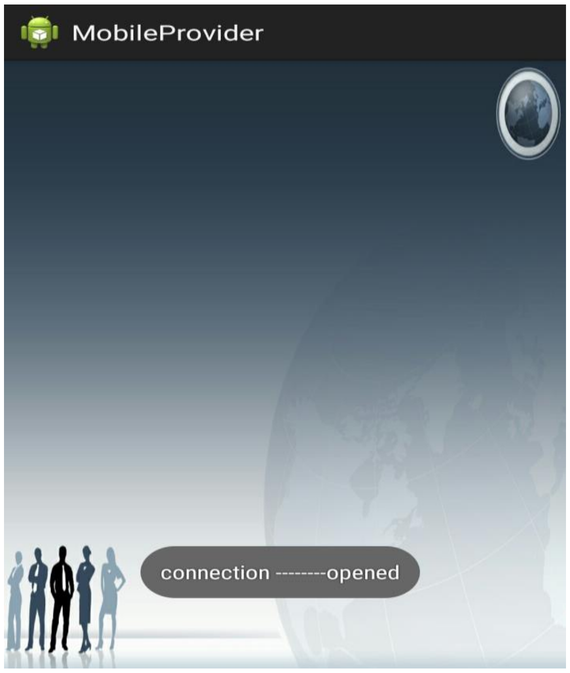


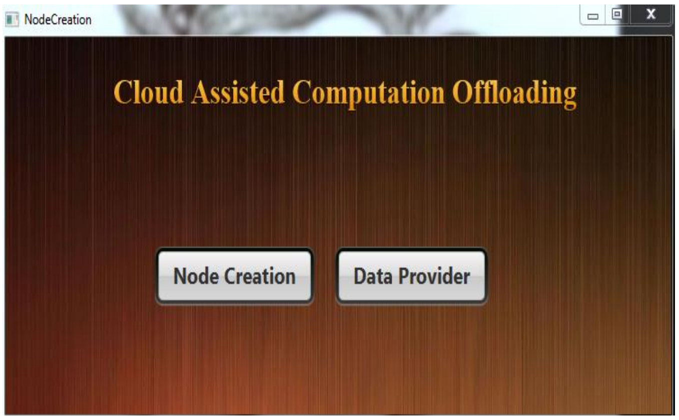


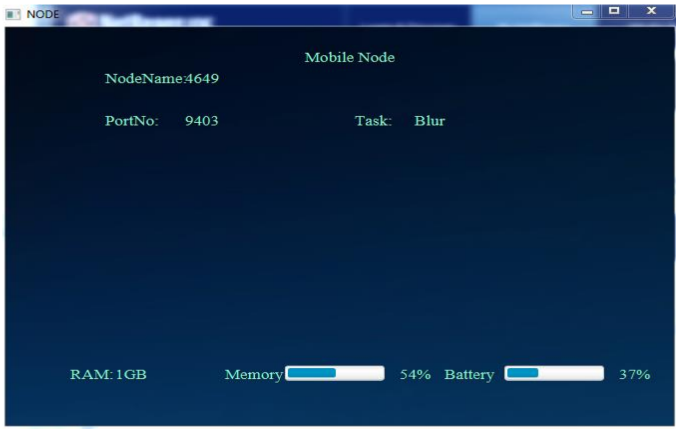

Mobile Cloud
User Login
UserName
jack
password
123
Ashok Nagar
Location
SignIn
SignUp

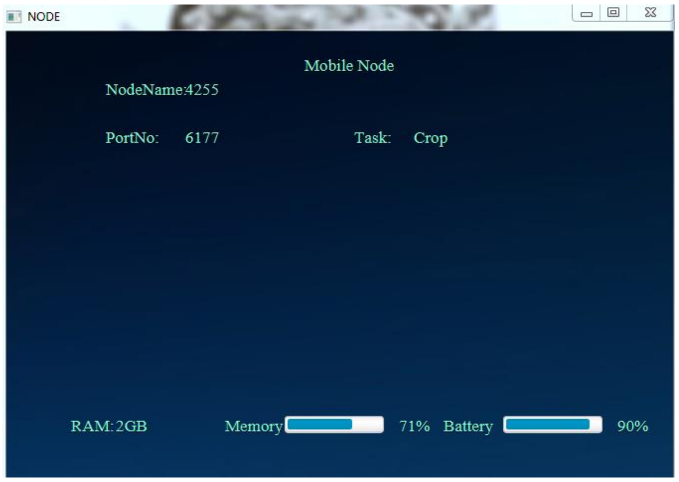

MobileProvider
Welcome To Mobile Provider
Submit


Mobile Cloud
Camera
Service Result
Output File
Exit


MobileProvider
connection --------opened







IV. FUTURE WORK

In our future work we will improve the intelligence of the Offloading framework by improving the heuristics and the addition of more context information, such that the estimation whether or not offloading is going to save energy or increase the computation speed will be more accurate. Although computation offloading can speed up computation and save energy, it is not guaranteed that it does. Another direction of our future work is to investigate which security measures need to be taken to secure the communication between the smart phone and the remote cloud resources. We also have to pay attention to the security implications of multiple users using a single remote resource, running foreign code on the remote resources, and making sure that remote services cannot disturb the working of other remote services.

V. CONCLUSION

In this paper we have presented a framework for computation offloading for smart phones, a recently rediscovered technique, which can be used to reduce the energy consumption on smart phones and increase the speed of compute intensive operations. The framework integrates with the popular open source Android framework and the Eclipse development tool. It provides a simple programming model, familiar to developers, that allows for a single interface with a local and a remote implementation.

REFERENCES

[1] 1. Dirk Bade, Gabriel Orsini, Winfried Lamersdorf " Context-Aware Computation Offloading for Mobile Cloud Computing : Requirements Analysis, Survey and Design Guideline" in *The 12th International Conference on Mobile Systems and Pervasive Computing*, (MobiSPC 2015).

[2] 2. Roelof Kemp, Nicholas Palmer, Thilo Kielmann and Henri Bal, "Cuckoo: a Computation Offloading Framework for Smart phones", in The Second International ICST Conference, MobiCASE 2010, Santa Clara, CA, USA, October 25-28, 2010.

[3] 3. Khadija Akherfi a, Micheal Gerndt a, Hamid Harroud b, "Mobile cloud computing for computation offloading: Issues and challenges", *Applied Computing and Informatics (2017).*

[4] 4. Khalid Elgazzar, Patrick Martin, and Hossam S. Hassanein, Senior Member, IEEE, "Cloud-Assisted Computation Offloading to Support Mobile Services", *IEEE transactions on cloud computing*, *vol.* 4, *no.* 3, July-september 2016.