Scientific Research Publishing

Int. J. Communications, Network and System Sciences, 2017, 10, 107-128
http://www.scirp.org/journal/ijcns
ISSN Online: 1913-3723
ISSN Print: 1913-3715

Achieving Mobile Cloud Computing through Heterogeneous Wireless Networks

Amani S. Alnezari[1], Nasser-Eddine Rikli[2]

[1]Community College in Alula, Taibah University, Madinah, KSA
[2]Department of Computer Engineering, King Saud University, Riyadh, KSA
Email: anazari@taibahu.edu.sa, rikli@ksu.edu.sa

How to cite this paper: Alnezari, A.S. and Rikli, N.-E. (2017) Achieving Mobile Cloud Computing through Heterogeneous Wireless Networks. *Int. J. Communications, Network and System Sciences*, **10**, 107-128.
https://doi.org/10.4236/ijcns.2017.106006

Received: April 15, 2017
Accepted: June 10, 2017
Published: June 13, 2017

Abstract

A Mobile Cloud Computing environment with remote computing and vertical handover capabilities will be considered. A fuzzy logic model will be suggested for both the offloading and handover functions using energy, delay and computation complexity as criteria. Based on the application requirements, some rules will be proposed and evaluated when deciding on the offloading then the selection of the network that keeps the traffic carried to the cloud at an acceptable level. All this ought to be achieved in real time while the application is still running. The proposed engine will be implemented and run on an Android mobile device connected to Google's App Engine servers. Results will be presented and analyzed, and conclusions will be provided.

Keywords

Mobile Cloud Computing, Vertical Handover, Fuzzy Logic

1. Introduction

Mobile Cloud Computing (MCC) builds on the concept of Cloud Computing with an added mobility feature [1]. It is considered as a viable solution to overcome the limitations of mobile devices, such as low computational power, limited battery life, and restricted memory capabilities. With the rapid advancements in telecommunications technologies, MCC has received attention as a way to overcome these limitations. In an MCC environment, the entire data is sent to the Cloud server from the mobile device, so that data may be processed using the powerful computing resources available in the Cloud servers. Afterwards, the mobile device gets the processed data back from the Cloud server, after all processes have been completed. It is such integration of mobile computing with Cloud computing that has become known by MCC.

MCC is expected to be used in heterogeneous networks that need different wireless network interfaces. Possible candidates to access the cloud may include WCDMA, GPRS, WiMAX, CDMA2000, and WLAN. Some issues arise as a consequence of the concurrent handling of the wireless connectivity and provisioning of the requirements of MCC applications (e.g., always-on connectivity, on-demand scalability of wireless connectivity, and the energy efficiency of mobile devices). The existence of heterogeneous networks increases the availability of Internet services and applications (*i.e.*, cloud services). Actually, most of the latest mobile phones have at least two interfaces to access these services either via WiFi or UMTS/HSPA [2], in addition to Bluetooth. An optimal use of heterogeneous networks with today's mobile phones includes seamless data transmission by using vertical handoff (VHO) solution.

Commonly, heterogeneous networks have different characteristics in network capacity, data rates, bandwidth, power consumption, Received Signal Strength and coverage areas. Vertical handoff is a process that will allow selecting the best network. It requires accurate and precise decisions about the availability of the networks and their resources for connection. The actual trend is to integrate complementary wireless technologies with overlapping coverage, to provide the expected ubiquitous coverage and to achieve the Always Best Connected (ABC) advantage. The ABC concept is to enable users to choose among a host of networks that best that suits his needs and to change when something better becomes available. It needs a framework for supporting mobility management, access discovery and selection, authentication, security and profile server [3].

Our focus in this study, will be based on reducing energy consumption and application execution time by providing seamless service through a combination of offloading of computing processes to the cloud and vertical handover between heterogeneous wireless networks [4]. The rest of the paper will be organized as follows. In the next section, the related work will be presented, followed by the proposed model. The simulation results will be presented an analyzed in the next section, and finally we will summarize our main findings in the conclusion section.

2. Related Work

In this section, we will present the latest major developments in designing energy-efficient techniques for the provision of services to mobile devices in a Cloud environment. The authors in [5] present an augmentation model called Cloud-based Mobile Augmentation (CMA). It leverages proximate and distant clouds to enhance, and optimize computing capabilities of mobile devices aiming at an execution of Resource-intensive Mobile Applications (RMAs). They then analyze the impacts of the distance between mobile cloud and some intermediate hops as influential factors on CMA performance, and show that there is a correlation between distance and intermediate hops on the overall execution costs of RMAs.

A mobile device perspective on energy consumption of applications is pre-

sented in [6]. It explores the impact of cloud-based applications on battery life of mobile devices. An algorithm "GreenSpot" is proposed that considers application features and energy-performance trade-off to determine whether cloud or local execution will be more preferable.

In [7], the authors provide an energy-efficient dynamic offloading and resource scheduling (eDors) policy to reduce energy consumption and shorten application completion time. The algorithm consists of sub-algorithms of computation offloading selection, clock frequency control and transmission power allocation. eDors algorithm was implemented in a real test bed and experimental results demonstrate that it can effectively reduce the energy consumption and application completion time, by taking advantage of the CPU clock frequency control in local computing and the transmission power allocation in cloud computing.

The research work in [8] presents a Mobile Augmentation Cloud Services (MACS) middleware that enables an adaptive extension of Android application execution from a mobile client into the cloud. Two prototype applications using the MACS middleware demonstrate the benefits of the approach. The evaluation shows that applications, which involve costly computations, can benefit from offloading with around 95% energy savings and significant performance gains compared to local execution only.

In [9], the authors present optimizing offloading strategies in Mobile Cloud Computing. They have developed a stochastic model to study the dynamic offloading in the context of MCC. The model captures various performance metrics and intermittently available access links (WLAN hotspots). It gives a new state and cost aware offloading policy that takes into account the mobility, the existing tasks, and the anticipated tasks.

The authors in [10] present a Mobile Cloud Computing Service in a heterogeneous wireless and mobile P2P network. They proposed an Integration between the two types of networks to bring the new concept in constructing mobile cloud computing system. The integrated network architecture provided the comprehensive infrastructure in enabling network as a service (NaaS) capabilities.

The objective of the work presented in [11] is to highlight the heavyweight aspects of current application offloading frameworks and to propose a lightweight model for distributed application deployment in MCC. The proposed framework eliminates the overhead associated with runtime-distributed deployment. It also provides a synchronization mechanism for coping with the issues of disconnections in the wireless network environment and ensuring consistency of distributed platform.

The authors in [12] describe the Phone 2 Cloud system, a computation offloading-based system for energy saving on smartphones in the context of mobile cloud computing. Additionally, it enhances the application's performance through reducing its execution time.

The authors in [13] present a novel offloading algorithm called "Dynamic Programming with Hamming Distance Termination" (DPH), which offloads as

many tasks as possible to the cloud when the network transmission bandwidth is high. This improves the total execution time of all tasks and minimizes the energy use of the mobile device.

The main objective of the work in [14] is the tradeoff between reducing execution time and saving energy consumption in cloud offloading systems. A novel adaptive offloading scheme is proposed and analyzed based on the tradeoff analysis.

3. Simulation Model and Algorithm

3.1. Basic Function

The main components of the considered mobile cloud computing model will consist of two parts. The offloading engine that will be used to test and evaluate the benefits of choosing to offload the application processes to the cloud, and the vertical handover function that will be used in the case of offloading selection. The handover function will continuously monitoring the possible communication link options to enhance the application performance and provide seamless service for mobile devices.

The criteria used in the first engine will be based on minimizing the energy consumption on the mobile device. The assistance of the cloud through offloading has the potential to save execution time and energy consumption. However, this energy savings should not exceed the time and energy cost due to the additional communication between mobile devices and cloud. Indeed, bulky data transmissions, especially under unfavorable wireless channel conditions, could consume a large amount of battery power as well. Hence, the offloading decision should be made for the components of the target application taking into account the current network conditions, the mobile device capabilities, and the application complexity.

In the second engine, the available wireless networks for cloud computing should be managed efficiently as well to support seamless services to mobile users regardless of their locations and movements. Here also, delays incurred during handover should be taken into consideration as well as the ping-pong effect due to short temporary variations in the network conditions.

3.2. Basic Operation

Figure 1 shows the basic architecture of the proposed model. The two engines are denoted by ODE (Offloading Decision Engine) and the HDE (Handoff Decision Engine). When the application starts, the application manager runs automatically ODE and contacts the cloud server. The basic purpose of this engine is to take the offloading decision that depends on many factors, such as mobile battery, wireless state, application requirement, to run the application locally on the mobile device or remotely on the cloud server. If the ODE decided to run the application remotely in the cloud, then it starts offloading the computing directly.

At the same time, the application manager runs the HDE to monitor wireless network status. When the existing connection is at stake, or when a new resource,

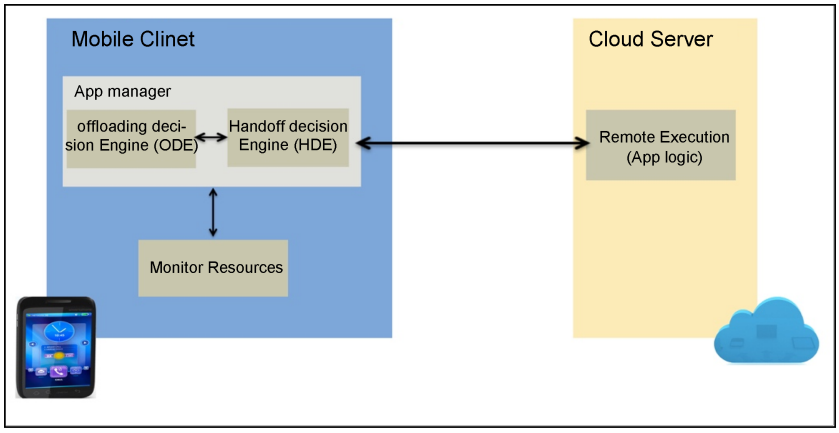


Figure 1. Overview of our model architecture.

which consumes less energy is found, a handoff request is initiated. The handoff is vertical handoff, between resources pertaining to different wireless networks.

In the case where there is no connection to the cloud or the ODE takes the decision to run the application locally, both engines will run in the background and take any decisions that may occur such as the connection to the cloud server is available, or there is a new wireless connection.

3.3. Choosing the Application

The initial idea was to create the application that has three different tests. Each test requires different computation levels. The reason for using three tests was to get a better understanding if different tasks could be more or less efficient for cloud offloading. We developed an Image processing application, which implements three filters. The low filter consists of 20 filters that are applied to a selected image. The medium filter has 40 filters and the high filter contains 60 different filters. Each of these filters needs different time and computation power to be applied. In addition, the image size will affect in time and computation consumed when applying the filtering.

The core functionality of the application is acquiring a user image on the Android client, apply image filters locally or on the cloud App Engine depending on a computed decision. Then displaying the image on the user's Android with filters applied. The theory of the application is to develop measure and compare the running of CPU-intensive processing on mobile and on cloud backbends.

We have conducted many experiments and tests by adding other options in the application such as execution locally, through the cloud or half local half cloud. Also, we took measurements of the execution time and energy in each test. By comparing the obtained results, we built an offloading decision engine.

This application is hosted on App Engine by using the App Engine backend. Google Cloud Endpoints consist of tools, libraries and capabilities. They generate APIs and client libraries from an App Engine application. The Endpoints provide a simple way to develop a shared web backend, which are used in developing the mobile applications. Because the API backend is an App Engine,

the app can uses all of the services and features available in the App Engine, such as Datastore, Google Cloud Storage, Mail, Url Fetch, Task Queues, and so forth [15]. The application uses (GCM) service to inform the user about which filter is being executed at that moment in the progress bar. The user will also be notified when a filter is finished or started.

3.4. Offloading Decision Algorithm

Fuzzy techniques are used to take the offloading decision. Fuzzy is chosen because the decision-making is fuzzy in nature. The service is offloaded when the need for offloading (Energy factor) is high and Time for offloading involved (Time factor) is low. Accordingly, the proposed algorithm calculates the Energy factor (Efactor) and Time factor (Tfactor) for offloading the execution.

The Energy factor is the measure for the need to offload the execution. This factor is calculated using two parameters E_{diff} and E_{level}. E_{diff} is the difference between energy consumption values for executing in the device and offloading it to the Cloud as shown in (1). The E_{level} represents the energy available in the mobile device now.

$$E_{diff} = \text{local Energy Taken} - \text{cloud Energy Taken}. \quad (1)$$

3.4.1. Energy and Time Consumption on Smartphone

In order to calculate the energy consumption consumed by running the application on the smartphone, we use Measurements-Activity function that measures the time and energy taken to perform local and cloud processing. To take energy measurement, it connects to a background service that measures the energy taken and calculates the values as shown in (2) and (3).

$$\text{local Time Taken} = \text{local End Time} - \text{local Start Time}. \quad (2)$$

$$\text{local Energy Taken} = \text{local End Energy} - \text{local Start Energy}. \quad (3)$$

3.4.2. Energy and Time Consumption on Cloud

Calculating energy consumption consumed by running the application on cloud (**cloud Total Energy Taken**) is more complicated than local (**local Energy Taken**). Before introducing how to calculate (**cloud Total Energy Taken**).

It needs three steps to finish computation offloading: First, upload the image to the cloud. Then, processing the image in the cloud and download image after processing to smartphone. Thus, (**cloud Total Energy Taken**) includes three parts: the energy consumed by upload "image" to the cloud (**cloud Upload Energy Taken**). The (**cloud Processing Energy Taken**), is the energy consumed while processing image in cloud and (**cloud Download Energy Taken**) the energy consumed by downloading results on smartphone.

$$\text{cloud Upload Time Taken} = \text{cloud Upload End Time} - \text{cloud Upload Start Time}. \quad (4)$$

$$\text{cloud Upload Energy Taken} = \text{cloud Upload End Energy} - \text{cloud Upload Start Energy}. \quad (5)$$

$$\text{cloud Processing Time Taken} = \text{cloud Processing End Time} - \text{cloud Processing Start Time}. \quad (6)$$

$$\text{cloud Processing Energy Taken} = \text{cloud Processing End Energy} - \text{cloud Processing Start Energy}. \quad (7)$$

$$\text{cloud Download Time Taken} = \text{cloud Download End Time} - \text{cloud Download Start Time}. \quad (8)$$

$$\text{cloud Download Energy Taken} = \text{cloud Download End Energy} - \text{cloud Download Start Energy}. \quad (9)$$

$$\text{cloud Total Time Taken} = \text{cloud Upload Time Taken} + \text{cloud Processing Time Taken} + \text{cloud Download Time Taken}. \quad (10)$$

$$\text{cloud Total Energy Taken} = \text{cloud Upload Energy Taken} + \text{cloud Processing Energy Taken} + \text{cloud Download Energy Taken} \quad (11)$$

If the difference between the energy consumption for offloading and executing to the device is high, the need for offloading is high. If the difference is low, the energy factor is low. For those values where the difference is neither low nor high, the energy level left in the device is used for evaluating the offloading factor.

Table 1 gives the values for quantifying the offloading factor. The table below explains the fuzzy logic rules that used to calculate the energy factor. Efactor level is assigned to make the decision more accurate. The table contains the values that effect the decision. Energy-Bigger Value represents the place that consumes more energy to execute the process. It could be a cloud or mobile device. The energy available in the mobile device now "E_{level}" has three levels: low, medium and high. Ediff represents the difference between the cloud-based and locally based. A function is used for calculating the level of Ediff. It returns the level whether low, medium, and high. In addition, it gives an integer number and maximum scale.

The number is compared with the scale. If the number is within the first third of the scale, the level is low. If the number is within the two-thirds of the scale, the level is medium. If the number is within the last third of the scale, the level is high. The last value represents the decision for the Efactor and its level. Time factor measures the time for offloading the execution to the Cloud. The time factor is related to the execution time (T_{rt}) for offloading and retrieving the results from the cloud and current received signal strength (RSS_{net}) of the wireless medium. The time factor for offloading is low, if the execution time is high, medium

Table 1. Decision making for energy factor.

Energy - Bigger Value	E level	Ediff	Efactor
Local	Low	Low or Medium or High	Cloud - High
Local	Medium	Low	Cloud - Low
Local	Medium	Medium	Cloud - Medium
Local	Medium	High	Cloud - High
Local	High	Low or Medium	Cloud - Low
Local	High	High	Cloud - Medium
Cloud	Low	Low or Medium or High	Local - High
Cloud	Medium	Low	Local - Low
Cloud	Medium	Medium	Local - Medium
Cloud	Medium	High	Local - High
Cloud	High	Low or Medium	Local - Low
Cloud	High	High	Local – Medium

or low, and the signal strength is low. However, the time factor is quantified as a medium, if the response time is not high, and the signal strength falls in the medium range. Moreover, the time factor for offloading is high, if the execution time is high, and the signal strength is medium. Table 2 gives the values for quantifying the Time factor. This quantification, along with the values of offloading factor help in deciding whether to offload or not.

The fuzzy logic rules are explained in the table above. They are used to calculate the time factor and assign Tfactor level to make the exact decision. The table contains the values that effect the decision. It begins with the Execution Time-Bigger Value, which represents the place that takes more time to execute the process; it could be a cloud or mobile device. Then, it includes the RSS that represents the received signal strength, which has three levels: low, medium and high. Moreover, it includes the ETdiff, which represents the difference in the Execution Time between cloud-based and the locally based. The ETdiff has three levels: low, medium and high. The last value of the table represents the decision for the Tfactor and its level.

Table 3 gives the decision for offloading based on the values of energy factor and Time factor. The service is offloaded when the need for offloading is high, and when the time for offloading is low. If the energy factor is neither high nor low, and the time factor is low, the decision of execution offloading to the cloud or not depends on the level of the factors.

3.5. Vertical Handover Engine

The (HDE) engine starts when the application starts execution. It monitors the network status and detects any change. If the connection with the current network is at stake, in order to improve its energy efficiency and reduce its latency,

Table 2. Decision making for time factor.

Execution time - Bigger Value	RSS	ETdiff	T Factor
Local	Low	Low or Medium or High	Cloud - Low
Local	Medium	Low	Cloud - Low
Local	Medium	Medium	Cloud - Medium
Local	Medium	High	Cloud - High
Local	High	Low	Cloud - Low
Local	High	Medium	Cloud - Medium
Local	High	High	Cloud - High
Cloud	Low	Low or Medium or High	Local - High
Cloud	Medium	Low	Local - Low
Cloud	Medium	Medium	Local - Medium
Cloud	Medium	High	Local - High
Cloud	High	Low	Local - Low
Cloud	High	Medium	Local - Medium
Cloud	High	High	Local - High

Table 3. Decision making for offloading.

Efactor Decision	Efactor Decision Level	Tfactor Decision	Tfactor Decision Level	Decision
Cloud	Low	Cloud	Low or Medium or High	Cloud
Cloud	Low	Local	Low	Preferably Local
Cloud	Low	Local	Medium or High	Local
Cloud	Medium	Cloud	Low or Medium or High	Cloud
Cloud	Medium	Local	Low	Cloud
Cloud	Medium	Local	Medium	Preferably Local
Cloud	Medium	Local	High	Local
Cloud	High	Cloud	Low or Medium or High	Cloud
Cloud	High	Local	Low or Medium	Cloud
Cloud	High	Local	High	Preferably Local
Local	Low	Cloud	Low	Preferably Local
Local	Low	Cloud	Medium or High	Cloud
Local	Low	Local	Low	Preferably Local
Local	Low	Local	Medium or High	Local
Local	Medium	Cloud	Low	Local
Local	Medium	Cloud	Medium	Preferably Local
Local	Medium	Cloud	High	Cloud
Local	Medium	Local	Low or Medium or High	Local
Local	High	Cloud	Low or Medium	Local
Local	High	Cloud	High	Preferably Local
Local	High	Local	Low or Medium or High	Local

the engine takes the necessary handoff decision to connect with a different network.

The decision-making algorithm decides whether to place the execution locally or remotely on the Cloud based on the number of interaction data transmitted per transaction and the current network status after applying the handoff. The engine works in the application background to takes necessary handoff. In addition, it updates the info of wireless network. Moreover, it informs the (ODE) with new information to take the accurate decision.

In the android platform, the automatic handover between 3G and WLAN networks is done usually when the current network link is going down. When the Android device connects to the Wi-Fi network, the platform automatically closes the 3G data connection. In contrast, when the Wi-Fi network is unavailable (or the user disconnects the Wi-Fi network from the device), the platform reactivates the 3G data connection.

When multiple access networks are available, the connection to a WLAN with strong signal can be used when the device is near the public access point, as the algorithm bellow. The handoff decision algorithm for the purpose of taking handoff decisions is shown in Table 4.

In our approach, received signal level of neighbor networks is periodically

Table 4. The handoff decision algorithm.

1: Mobile device connect to access point that Connected with Cloud.
2: While
3: Measure RSS_{new}; and BR_{new} of discovered access point
4: if (RSS_{new} -$RSS_{current}$ > $RSS_{Threshold}$) and(BR_{new} > $BR_{current}$) then
5: $T_{current}$t = N * (D / $BR_{current}$t + T_{prop} + ET_m + ET_{cloud})
6: T_{new} = N * (D / BR_{new} + T_{prop} + ET_m + ET_{cloud}
7: if (T_{new} < $T_{current}$) then
8: handoff occur (RSS_{new} = $RSS_{current}$, BR_{new} = $BR_{current}$)
9: take new decision for offloading
10: end if
11: end while

measured, and the "best" network (in terms of higher signal level) is selected as the current access network.

4. Performance Analysis

In this section, we introduce the results of our application to achieve mobile cloud computing. For comparison purposes, three kinds of processing (low, medium and high) filters run remotely in Google app engine as cloud side and runs locally in android device as client side. Also, we made tests in different input size, bandwidth, delays and mobile devices. Both static and dynamic code offload is tested and recorded results in order to measure the cost of offloading decision.

The execution time is measured on the Android mobile device, and Google's App Engine servers. The energy consumed is measured by comparing the energy consumed when running the app locally, and when running it remotely. These measurements will provide means to analyze the viability of mobile cloud computing, and evaluate whether executing the code remotely on more powerful servers is advantageous or not. The time needed to communicate with the remote servers is measured to analyze the communication added costs of the remote execution. Furthermore, with the level and complexity of processing, the measurement of time is important in the terms of user experience and application performance. The energy consumption and execution time of three types of processing are evaluated; as shown in Table 5 with respect to many factors. The table examines how these factors affect the energy consumption and execution time of the applications. The table evaluates the influence of each specified factor on both the energy consumption and execution time of three applications under different ranges. The results obtained from the experiments are shown below with the focus on the execution time and power consumption (The Table 6 distinguishes the image sizes that used in all tests).

4.1. Comparison of the Cloud-Based and Local Based Test Results

Figure 2(a) shows the execution time of a low processing case under different input sizes and network status. In all cases, the value of cloud is more than the value of smartphone. The gap between them expands as the input grows. In the first test, the difference between the cloud-based and locally-based is a little bit. However, in the second test, the input size value is 3 MB, which is almost the

Table 5. Applications used in experiment.

NO	Application	Description
1	Low CPU-intensive processing	It consists of 10 simple filters that Applied in selected images
2	Medium CPU-intensive processing	It consists of 40 middle filters that Applied in selected images
3	High CPU-intensive processing	It consists of 100 complex filters that Applied in selected images

Table 6. Sizes of the images used in tests.

Image	Size
Small image	50 KB
Medium image	3 MB
Large image	6.5 MB

doubled value. Finally, the last test is almost three times the perfect time with the worst network connection. The reason lies in the additional communication cost of the cloud and smartphone, which surpasses the processing cost. Therefore, the app should run locally whatever the status of network in the low processing option.

The execution time of medium processing filters is measured. In addition, this work studies how the execution time in medium case depends on the application input parameters and different networks situation. It is clear that running the application on the cloud does not always consume more time than running it locally. That is shown in Figure 2(b). The reason is the execution time for medium processing on the mobile will cost time. It will grow as the image size is increased. Moreover, the gap between running the application on the cloud and locally becomes smaller. In addition, it is reflected for the cloud. It is obvious from the Figure that the execution time in cases of medium or large image sizes, with a good connection, will cost less in the cloud. Regarding this option, if the input size is more than 3 MB, with an ideal network available, the app should always run remotely.

Figure 2(c) shows the execution time of high processing case under different input sizes and network status. In all cases, the execution time values on the cloud are less than the execution time values on the smartphone. The gap between them expands, as the input grows. The gap reaches up to half time with the large input sizes. In the first test, the difference between the cloud-based and locally based is a little bit. In the second test, the difference is less with the high-speed network, and a little bit more with the slow network connectivity. Finally, in the last test, when the input size is more than 6 MB, running the application on the cloud will takes almost half time. That is compared to the locally based on the perfect network case. However, with the worst network connection, it takes less time than locally. Therefore, the app should always run remotely on the cloud, whatever the status of network in the high processing case.

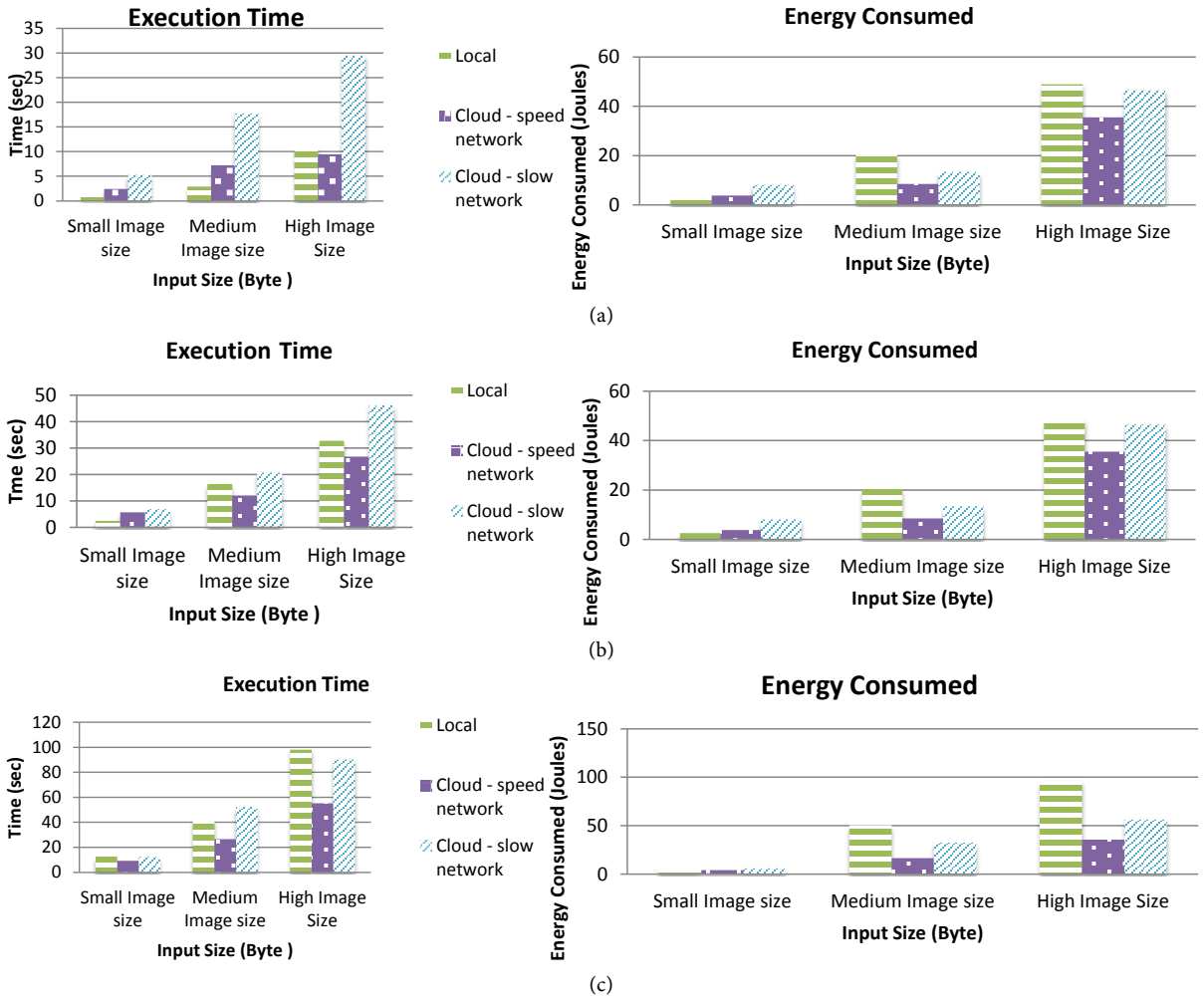


Figure 2. Execution time and energy consumption for different processing kind.

The results of power consumption are shown in the second row of the Figure 2. It is clear in the Figure 2(a) that the difference in energy consumption between the mobile and cloud will vary. Since it depends on the image size and network status. In the case of small image size, the difference in energy consumption between the mobile and cloud is very small. However, the difference is increased in the locally based for the slow network. The same situation happens with the medium image size. But the difference is increased in the slow network.

Finally, the energy consumed in the cloud is less than the energy consumed in the mobile in the large image size. In addition, the cloud will consume more energy, if the network is in its worst case. Therefore, the app should run locally and remotely depending on the status of network in the low processing option.

Figure 2(b) illustrates the energy consumption of medium processing under different input size. In the first test, it is obvious that the energy consumed is smaller in the locally based device, with the small input sizes only that are equal to "50 KB". In other cases, the energy consumed is smaller on the cloud-based device. Regarding the medium processing option, running the application on the cloud will save the energy consumed whatever the network status. It is noticea-

ble in Figure 2(c) that the energy consumed will be less if the app is running on the cloud whatever the input size and network status. However, with the high computing applications, running the application on the cloud will cost the mobile a little energy compared to running it locally. That is because the high intensive processing in the mobile device consumes resources. In addition, it requires much RAM, memory, and energy.

4.2. Effects of Upload, Download and Process Ratio

The objective of this section is to analyze the performance of running the application remotely under different network situation. The application is tested in different latencies, bandwidth available and response times. The effect of bandwidth available will lead to delay and packet loss. The results of this section are that the offloading decision should measure the network condition constantly as well as estimate the bandwidth available and latency.

4.2.1. Case of Low Processing, Low Image

The test is performed on three-network status. The first network bandwidth available is 5.88 Mbps. Moreover, its latency is almost 20 ms. The second network has more latency, about 35 ms. In addition, its bandwidth available is 2.50 Mbps. The third network has the worst situation. It has a low bandwidth, which is 1.05 Mbps. Also, it has a low delay, which is 18 ms approximately.

The execution time consists of three parts as shown: Upload time, Processing time and Download time. The Figure 3(a) presents the changes in execution time with the different network status. It is clear that the best performance is in the first network. The second network has overhead in processing time due to delay rate. In addition, it has a medium upload ratio. Finally, the third network has the worst performance, although it has the lowest delay. However, with a low bandwidth, the upload and download cost a lot of time.

The energy consumption of the application, which is running on the cloud is influenced by the bandwidth available and delay in different networks, as shown in the Figure 3(b). An Upload energy, Processing energy, and Download energy compose the total energy. In the first network, the power consumption on the cloud is decreased because the bandwidth available is increased and the delays is decreased. The second network has more power consumption than the first one due to the small bandwidth available and delay rate (35 ms). Finally, the third network has the biggest energy consumption value, because the bandwidth available is decreased until (1.05) Mbps.

4.2.2. Case of Medium Processing, Medium Image

The application is tested on three networks status. The first network bandwidth available is 5.88 Mbps, and its latency is almost 20 ms. The second network has the worst situation. Its latency value is 50 ms, and its bandwidth available is 2.00 Mbps. The third network has the standard situation. It has a low bandwidth available, which is 2.50 Mbps. In addition, it has the lowest delay, which is 30 ms approximately.

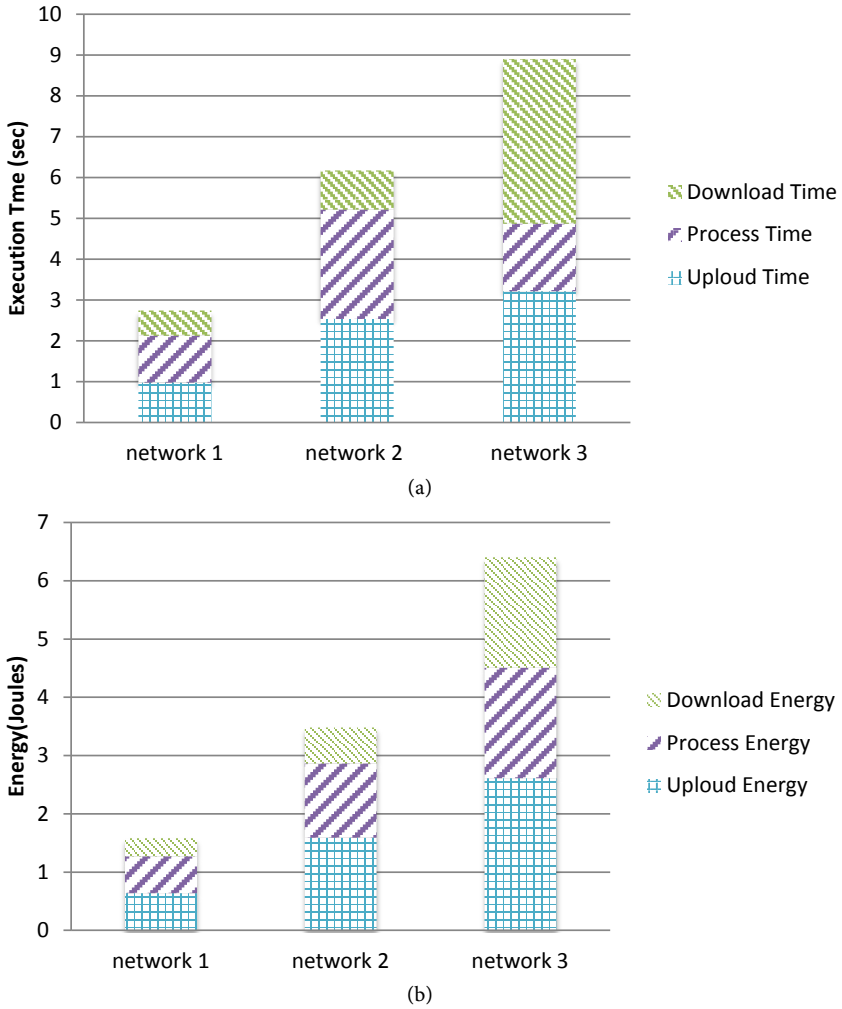


Figure 3. Execution time and energy consumption for low processing under different networks.

Figure 4(a) describes how the bandwidth available and delay effect in running the medium processing apps on the cloud under different network status. It is obvious in the Figure that the execution time is very fast at the first network compared to the other networks. The application achieves the best performance due to the high bandwidth available and the low latency. The application achieves the worst performance with the second network due to the high delay, which effects the processing time in the cloud, in addition to the medium upload ratio. Finally, the application in the third network has an acceptable performance. It has a medium delay and bandwidth. However, the upload costs a bit of time.

On the other hand, the energy consumed by the medium processing application on the cloud is increased or decreased. It depends on the bandwidth available and delay ratio in different networks, as shown in the **Figure 4(b)**. It is noticeable that the energy is saved significantly in the first network. The second network has the largest value of the energy consumption due to the small bandwidth available and high delay. Finally, the third network has the acceptable value

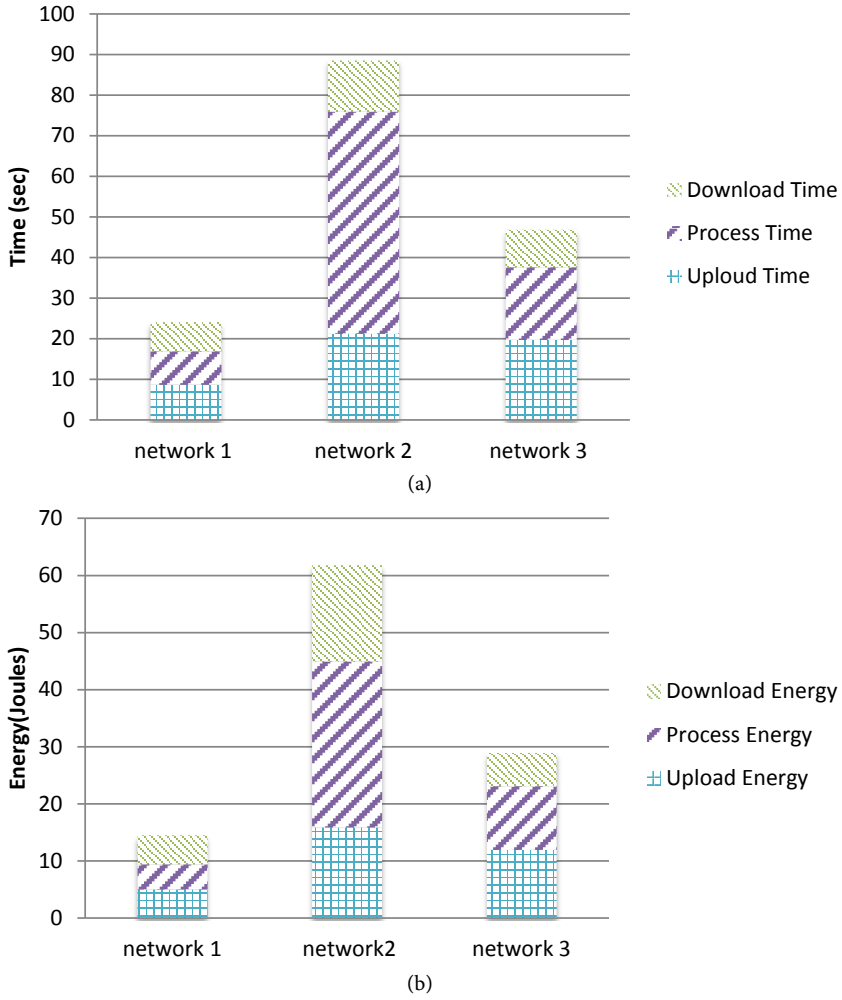


Figure 4. Execution time and Energy consumption for medium processing under different networks.

of energy consumption compared to the other networks status.

4.2.3. Case of High Processing, High Image

The test is performed to three-network status. The first network has a low bandwidth available, which is about 1.00 Mbps. In addition, its latency is almost 35 ms. The second network has more latency, which is about 25 ms. In addition, its bandwidth available is 5.00 Mbps. The third network has a high bandwidth, which is about 5.05 Mbps. Moreover, its delay is 30 ms approximately.

In most cases in the high processing apps, as proved before, the value of execution time on the cloud is always less than value of execution time on the smartphone. Therefore, the total time, as shown in the **Figure 5**, is effected by the network status, which has different bandwidth available and delay. In the case of high processing apps with the large image, the effect of input size has a great impact on the offloading process, because of the large amounts of data transfer between the mobile and cloud. In addition, the complexity of computing on the high intensive processing filters requires offload computing on the cloud server.

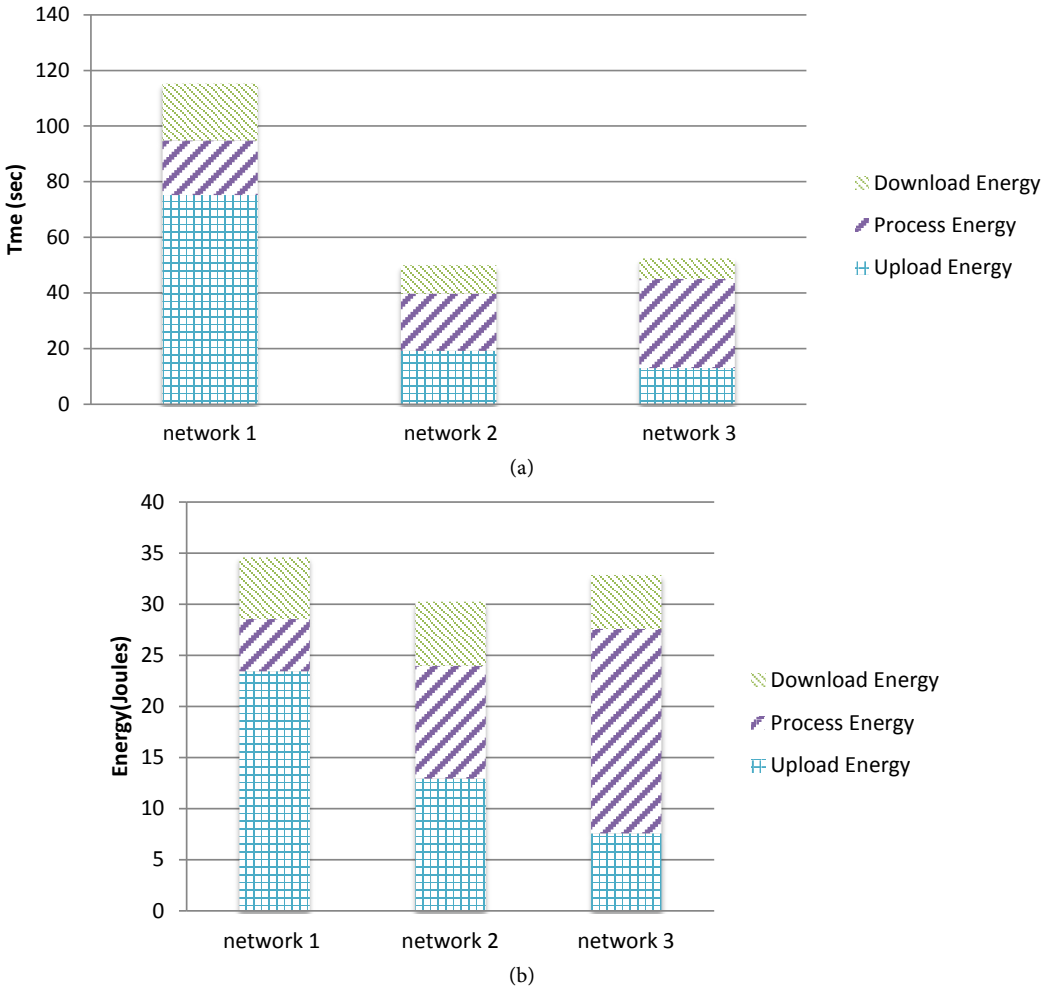


Figure 5. Execution time and Energy consumption for high processing under different networks.

These types of applications obviously require more bandwidth available and low latency, because the response time is effect by the both factors. The power consumption on the cloud is decreased in the high processing applications for most networks. However, the total value is effected by the current situation of network.

4.3. Comparing the Decision Engine Based Test Results

This section shows the results of decision engine, with the effect of three factors; including the input size, bandwidth available and CPU processor speed. Some factors related to the application, wireless status, and mobile device specifications.

4.3.1. Input Size

Figure 6 shows the execution time and power consumption of the low CPU-intensive processing under different input sizes. The red line in the Figure represents the execution time and power consumption, which depends on the decision engine. The green line represents running the application locally on the smartphone. The purple line represents running the application remotely on the cloud.

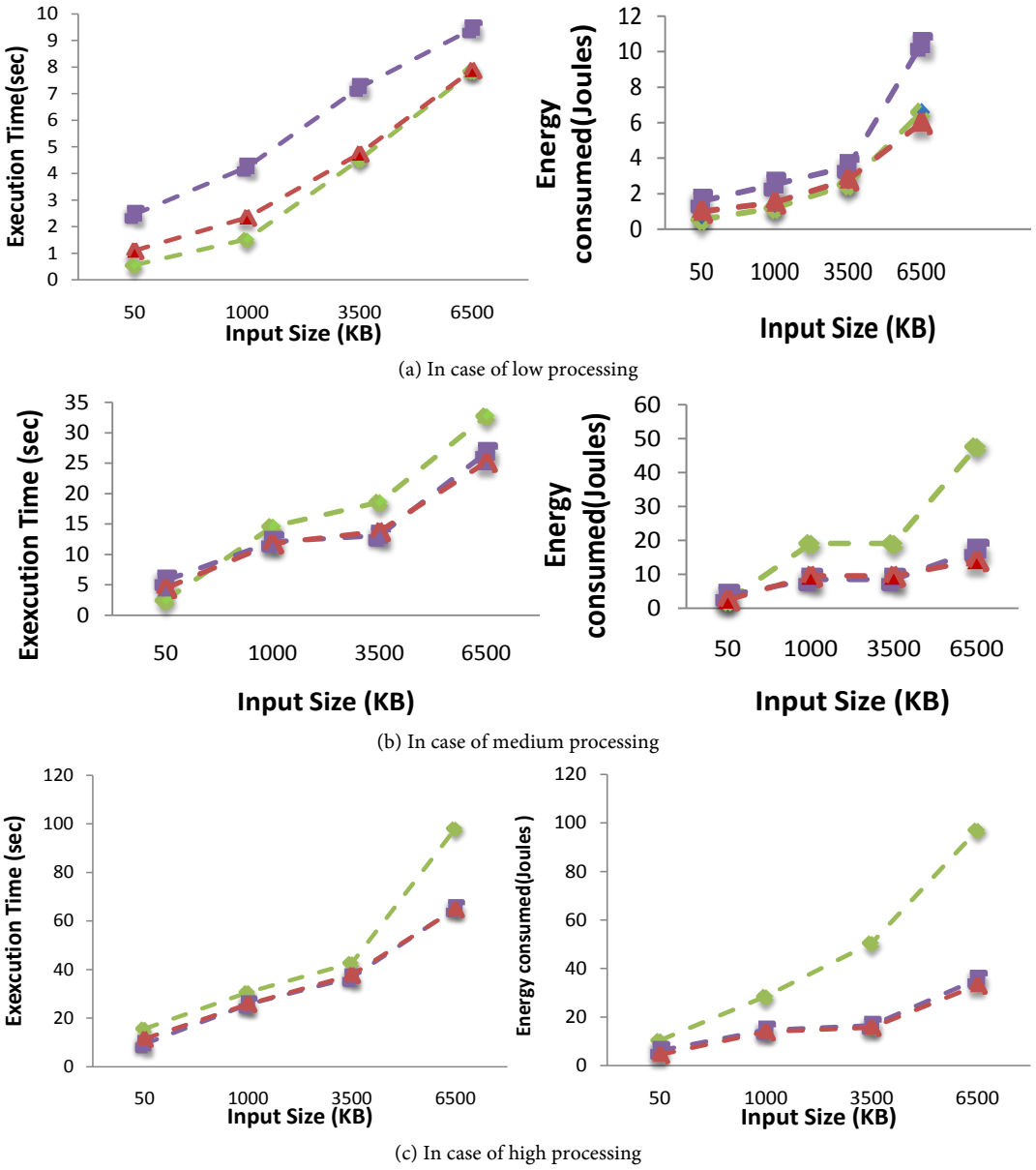


Figure 6. Execution time and energy consumption of three applications under different input size.

First, it is clear in the case of low processing that the value of the execution time when the app is running on the cloud is more than the value of the execution time when the app is running locally. The gap between them expands on the input grows. Because the time cost due to the data transmission between the cloud and mobile device is more than the time cost of running the app locally. Therefore, regarding this kind of processing, the application should run on the mobile device. In this case, the decision engine can make a wise offloading decision. The energy consumed by the low CPU-intensive processing running on the cloud is much more than running on the smartphone. The reason is that the energy consumed by applying the filters on the mobile device is less than the energy consumed by applying the filters locally due to the data transmission, which includes sending input data and receiving results. It is obvious that with

the low CPU-intensive processing application, it is better to run it locally.

In the medium processing case, it is noticeable that the engine takes the accurate decision as shown in Part B of the Figure 6 Because the execution time of applying filters on the cloud is less than the execution time on a mobile device when the input size is larger than 1 MB. So, it is recommended to run the application on the cloud in that case. The results in Part B illustrates that the energy consumed by the medium CPU-intensive processing is increasing when the input size is smaller than 1 MB. Since the application that runs on the cloud costs more energy than the application that runs locally. The power consumption on the smartphone is larger than the power consumption on the cloud. Whenever the input size is increased, the cost of processing is increased, too. Therefore, processing in the cloud is better because the cost of processing is more than the cost of transmission data.

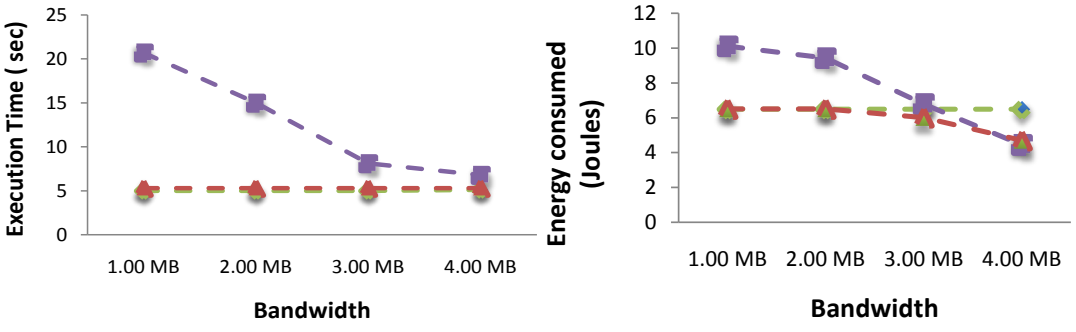
Furthermore, the costs of execution time on the cloud are less than the costs of execution time on the smartphone at the high processing as shown in Part C of the Figure 6 As the input size is increased, the engine decides to run the app remotely on the cloud, especially when the input size is greater than 1 MB. However, the power consumption of running high CPU-intensive processing on the mobile device is much more than the power consumption on the cloud as shown in Part C of the Figure 6. The power consumed due to the processing is much more than transmitting data, because of the high complexity of filters on the mobile device. It is observed that with such kind of the application, the decision should always be offloaded on the cloud. Moreover, it is obvious that the engine saves much more energy at the high CPU-intensive processing under these circumstances. As it is noticeable in the all cases, the engine takes the decision to run the app locally or on the cloud based on the best results. In most cases, the engine takes the same value or a very close value to the best performance. So, the decision engine makes a wise decision, which improves the user's experience.

4.3.2. Bandwidth

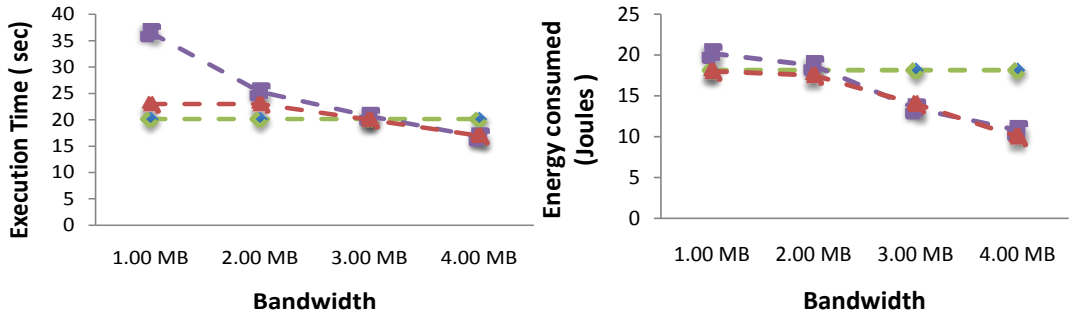
The second factor is associated with the effect of wireless network status. The bandwidth available is an important factor. Since it directly affects the performance of app, especially when it varies from one network to another in the heterogeneous environments. This project studies this factor and its effect to the engine and performance of app.

The first column in the Figure 7 shows the relationship between the execution time of three applications and bandwidth. The second column represents the energy consumption of three applications under different bandwidth. In the first case is a low processing case as shown in Part A. It shows that running the app on the cloud always takes more time than running it locally. The execution time is decreased gradually as the bandwidth available is increased. Therefore, the decision engine can make a wise offloading decision for this kind of processing by running the app on the mobile device.

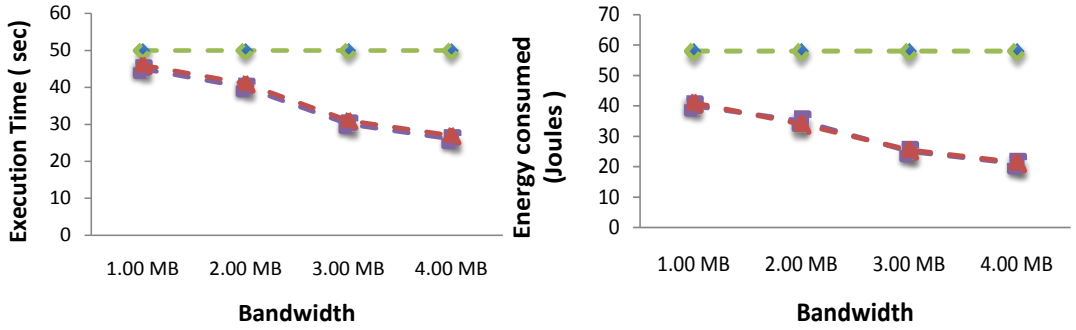
In the same case, the value of power consumption on the cloud is always



(a) In case of low processing



(b) In case of medium processing



(c) In case of high processing

Figure 7. Execution time and energy consumption of three applications under different bandwidth.

higher than the value of power consumption on the mobile device. Therefore, it is better to run this kind of apps locally. The explanation for this is that the energy consumed by the processing computation on the mobile device is less than the energy consumed by the data transmission. However, the energy consumption whether remotely or locally is getting closer and closer as the bandwidth available is increased.

In the medium processing case, as in Part B of the Figure 7, the results are little different. The execution time on the cloud is less than the execution time on the mobile device when the bandwidth available is more than 3.5 MB/s.

The opposite occurs when the bandwidth available is smaller than 3.5 MB/s. Regarding the energy consumption of app on the cloud and mobile device, the decision engine offloads processing to the cloud, and it makes the right decision again. The CPU-intensive processing spends more time on the mobile device than on the cloud as shown in Part C of the Figure 7.

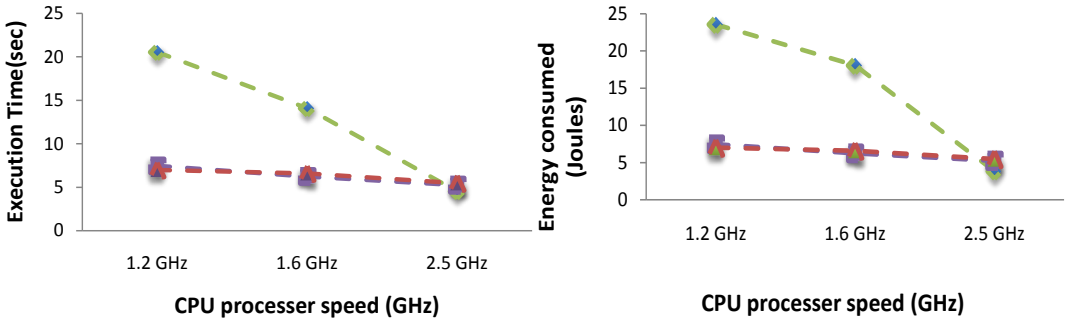
Furthermore, the power consumed on the mobile device is kept the same. It is

clear in the Figure 7(b) and Figure 7(c) that the power consumption on the mobile device is much more than the power consumption on the cloud for the medium and high CPU-intensive processing running under such circumstances. In addition, the power consumption on the cloud is decreased as the bandwidth available is increased. So, it is recommended to offload the apps on the cloud because that saves much energy for users. The engine runs the app locally or on the cloud, and makes the right decision to get the best results.

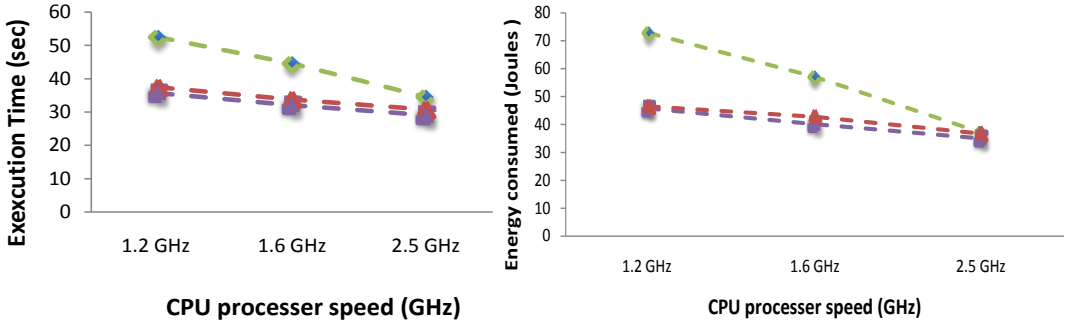
4.3.3. CPU Processor Speed

The last factor is related to the capabilities of mobile device. The smartphone specifications play as an important factor, since they directly affect the performance of applications in supporting the connection with any wireless networks.

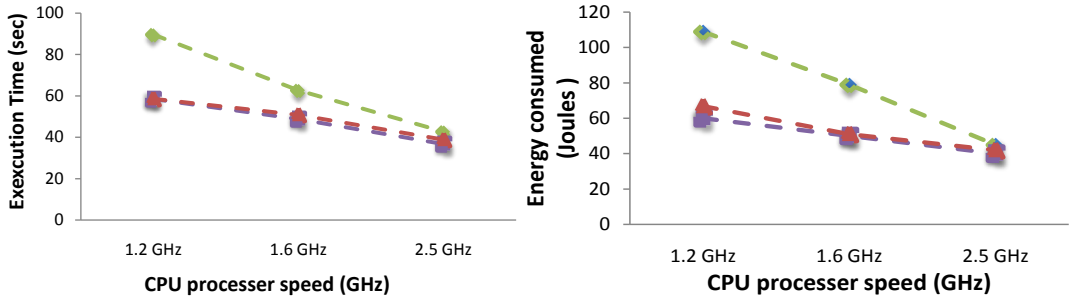
Figure 8 shows the relationship between the execution time and energy consumption of the three applications and CPU processor speed. The first case, Part A, represents a low processing case. In this case, running the app on the cloud always takes less time than running it locally. The execution time on the cloud



(a) In case of low processing



(b) In case of medium processing



(c) In case of high processing

Figure 8. Execution time and energy consumption of three applications under CPU processor speed.

almost has a constant value, because the cloud capabilities do not change or effect by the smartphone specifications. The execution time on the locally based device is decreased gradually as the CPU processor speed is increased; until it reaches a limiting value with the fast processors. Therefore, the decision engine can make a wise offloading decision for this kind of processing by offloading to the cloud.

The medium and high processing have the same results as shown in Part B and C of the Figure 8. The execution time on the cloud is less than the execution time on the mobile device. The gap between them is decreased as the CPU processor speed is increased. Therefore, the decision engine can make a wise offloading decision by offloading to the cloud for all the cases. The engine always offloads the apps to the cloud for all the cases as shown in the Figure 8. The value of power consumption on the cloud is always smaller than the value of power consumption on the mobile device. In addition, the energy consumption of running the apps on the cloud is almost not changed. It is worthy to say that for this kind of apps, the decision should always be running the apps remotely

5. Conclusion

In conclusion, this project shows obvious advantages of the mobile cloud computing technology. In addition, by applying engines to decide the offloading and vertical handover, that will improve both the application execution time and the energy consumed by the mobile device. These results prove that the cloud computing is very probable, and the offloading computation to the cloud server is a viable timesaving option. As long as the network speeds are suitable. It is an advantage to offload the computationally intensive applications to a more powerful server. It is not only an advantage, but it is also necessary in some situations. E.g., it is necessary when the mobile device is unable to run certain applications due to memory restrictions or limited mobile specifications. Generally, the most cloud platforms show the advantage of offloading the applications to the cloud resources in the context of providing a SaaS. By outsourcing computation offloading to the backend servers, the simple mobile device becomes more powerful. However, there is no best or simple implementation of the mobile cloud computing. Options include dynamic vs. static code offload, method vs. OS migration, and various connections protocols. Different applications have different resource requirements effecting the best possible connection to the cloud. Finally, the MCC application should be built to adapt intelligently to different changes in the surrounding networks, device capabilities and application requirements. That is necessary to make the device decides the best particular application for it.

References

[1] Song, W. and Su, X. (2011) Review of Mobile Cloud Computing. 3*rd International Conference on Communication Software and Networks*, Xi'an, 27-29 May 2011, 1-4. https://doi.org/10.1109/iccsn.2011.6014374

[2] Dinh, H.T., *et al.* (2013) A Survey of Mobile Cloud Computing: Architecture, Ap-

plications, and Approaches. *Wireless Communications and Mobile Computing*, **13**, 1587-1611. https://doi.org/10.1002/wcm.1203

[3] Akhila, S., *et al*. (2012) An Overview on Decision Techniques for Vertical Handoffs across Wireless Heterogeneous Networks. *International Journal of Scientific & Engineering Research*, **3**, 1-6.

[4] Sanaei, Z., *et al*. (2014) Heterogeneity in Mobile Cloud Computing: Taxonomy and Open Challenges. *Communications Surveys & Tutorials*, **16**, 369-392. https://doi.org/10.1109/SURV.2013.050113.00090

[5] Abolfazli, S., *et al*. (2014) An Experimental Analysis on Cloud-Based Mobile Augmentation in Mobile Cloud Computing. *IEEE Transactions on Consumer Electronics*, **60**, 146-154. https://doi.org/10.1109/TCE.2014.6780937

[6] Namboodiri, V. and Toolika, G. (2012) To Cloud or Not to Cloud: A Mobile Device Perspective on Energy Consumption of Applications. *IEEE International Symposium on World of Wireless, Mobile and Multimedia Networks*, San Francisco, 25-28 June 2012, 1-9. https://doi.org/10.1109/wowmom.2012.6263712

[7] Guo, S., Xiao, B., Yang, Y. and Yang, Y. (2016) Energy-Efficient Dynamic Offloading and Resource Scheduling in Mobile Cloud Computing. 35*th Annual IEEE International Conference on Computer Communications*, San Francisco, 10-14 April 2016, 1-9.

[8] Kovachev, D. and Klamma, R. (2012) Framework for Computation Offloading in Mobile Cloud Computing. *International Journal of Interactive Multimedia and Artificial Intelligence*, **1**, 6-15.

[9] Hyytiä, E., Thrasyvoulos, S. and Jörg, O. (2013) Optimizing Offloading Strategies in Mobile Cloud Computing.

[10] Larosa, Y.T., *et al*. (2011) Mobile Cloud Computing Service Based on Heterogeneous Wireless and Mobile P2P Networks. 7*th International Wireless Communications and Mobile Computing Conference*, Istanbul, 4-8 July 2011, 661-665.

[11] Shiraz, M., Gani, A. and Khokhar, R.H. (2012) Towards Lightweight Distributed Applications for Mobile Cloud Computing. *IEEE International Conference on Computer Science and Automation Engineering*, Zhangjiajie, 25-27 May 2012, 89-93.

[12] Xia, F., *et al*. (2014) Phone2Cloud: Exploiting Computation Offloading for Energy Saving on Smartphones in Mobile Cloud Computing. *Information Systems Frontiers*, **16**, 95-111. https://doi.org/10.1007/s10796-013-9458-1

[13] Shahzad, H. and Szymanski, T.H. (2016) A Dynamic Programming Offloading Algorithm for Mobile Cloud Computing. 2016 *IEEE Canadian Conference on Electrical and Computer Engineering*, Vancouver, 15-18 May 2016, 1-5. https://doi.org/10.1109/CCECE.2016.7726790

[14] Wu, H., Wang, Q. and Wolter, K. (2013) Tradeoff between Performance Improvement and Energy Saving in Mobile Cloud Offloading Systems. *Communications Workshops*, Budapest, 9-13 June 2013, 728-732.

[15] Google Developers. What Is Google App Engine. https://developers.google.com/appengine/docs/whatisgoogleappengine