


RESEARCH

Open Access



# Efficient resourceful mobile cloud architecture (mRARSA) for resource demanding applications

Asharul Islam<sup>1\*</sup> , Anoop Kumar<sup>1</sup>, Khalid Mohiuddin<sup>2</sup>, Sadaf Yasmin<sup>3</sup>, Mohammed Abdul Khaleel<sup>2</sup> and Mohammad Rashid Hussain<sup>2</sup>

## Abstract

For mobile clients, sufficient resources with the assurance of efficient performance and energy efficiency are the core concerns. This article mainly considers this need and proposes a resourceful architecture, called mRARSA that addresses the critical need in a mobile cloud environment. This architecture consists of cloud resources, mobile devices, and a set of functional components. The performance efficiency evaluates implementing the proposed context-aware multi-criteria decision offloading algorithm. This algorithm considers both device context (network parameters) and application content (task size) at run time when offloading an executable code to allocate the cloud resources. The appropriate resources select based on offloading decisions and via the wireless communication channels. The architecture's remarkable component is the signal strength analyzer that determines the signal quality (e.g. -60 dBm) and contributes to performance efficiency. The proposed prototype model has implemented several times to monitor the performance efficiency, mobility, performance at communication barriers, and the outcomes of resource-demanding application's execution. Results indicate performance improvement, such as the algorithm appropriately decides the cloud resources based on device network context, application content, mobility, and the signal strength quality and range. Moreover, the results also show significant improvement in achieving performance and energy efficiency. Sufficient resources and performance efficiency are the most significant features that distinguish this framework from the other existing frameworks.

**Keywords:** Mobile cloud architecture, wireless communication channel, signal strength analyzer, cloud service provisioning, energy efficiency, executable code switching

## Introduction

Today, energy augmentation and resourceful computation ability are increasingly the major concerns by the Smart Mobile Device (SMD) users. Intelligent wireless communication technologies, such as SMDs, smart-watches, tablets, and other multipurpose tools, have become an inseparable part of human life [1]. Growing applications' execution on resource-limited SMDs determine the necessity of Mobile Cloud Computing (MCC) for the resourceful execution environment [2]. SMDs or mobile devices are limited in computing resources (CPU, memory size, disk capacity, processing power), and efficiency-enhancing usability features (battery life,

size, weight) make mobile devices resource-constraint [3]. Mobile devices integrated with cloud offer a sizeable virtual environment on a physical layer, i.e., MCC architecture. Several MCC architectures develop [4] to augment mobile device's resources and provide a web-based efficient platform to execute intensive applications in a "pay-as-you-consume" model. Cloud resources use to augment mobile device resources and to overcome resource limitations. By using MCC, many intensive applications discussed in [1, 5], which require augmented resources (computing speed, RAM, higher disk capacity) to execute efficiently.

Although many researchers have paid attention to incorporate high-performance core processors into mobile devices to bridge the gap between the device resources' abilities and increasing usage by executing intensive

\* Correspondence: [ashar.islam@gmail.com](mailto:ashar.islam@gmail.com)

<sup>1</sup>Banasthali Vidyapith, Niwai, Rajasthan, India

Full list of author information is available at the end of the article

applications. They pay attention to different layers of MCC architecture. At the physical layer, several architectures (Cloud computing with mobile terminals, Virtual cloud computing provider, Cloudlet, and Operator Centric Mobile Cloud Architecture (OCMCA)) [4] are the state-of-the-art. Moreover, physical layer architecture takes care to connect the mobile devices to the cloud network and ensures both user mobility and confidentiality. At the application layer, the aim is to develop new software solutions that improve application execution performance in MCC architectures. Further, MCC architectures consider for controlling and monitoring the consumption of device resources. Nevertheless, they shall not be included for the performance measures as these architectures are complementary to the physical layer architectures [4].

MCC architectures significantly consume on-demand cloud services and augment device resources, such as computation power, shared memory, storage, and network access with various offloading frameworks. These frameworks are the methods of offloading resource-intensive portions of running mobile applications to the cloud, which sends the required results to the mobile client after performing intensive computations [3]. The whole process of offloading augments device resources, and conserves its resources (Dolphin, <https://www.dolphin.com/>; CloudMagic). MCC architectures with cloud associated services design to overcome mobile device limitations and transformed into augmented mobile platforms. Hence, the architectures and offloading frameworks widely deployed for resource-intensive mobile applications. However, offloading mechanisms for mobile applications still inherent own limitations, such as WAN latency is a big concern in leveraging cloud resources in a mobile environment [6]. WAN links, 4G, LTE, & Wi-Fi minimize the issues of connectivity, latency, and bandwidth. Even with the introduction of 5G should optimize connectivity performance and enrich user experience. The wireless radio link issues such as signal loss, noise, radio link wake-up delay should be the concerns in leveraging cloud services.

There are many established offloading schemes, discussed in [1–3, 7] which considered application partitioning and offloading the resource-intensive tasks to the cloud is the practical solutions at the application layer. These offloading schemes alleviate the pressure of application execution in mobile devices [1]. Generally, in offloading schemes, the executing application is divided into two parts: resource-intensive part offloads to a remote cloud via the wireless channel, and the other part necessarily runs in the local device. These schemes focus all most all the components of MCC architectures at both layers. A recent study [3] uses Cloudlet and provides a taxonomy of solutions at the application layer

within the proximity of mobile devices [7]. focusses on the performance of mobile applications considering offloading adoption algorithms [1]. Proposes the taxonomy of enabling techniques at the application layer and classifies various established offloading schemes considering the physical layer. The offloading schemes augment device resources and leverage cloud resources efficiently for the resource-intensive services. The study's approach describes in "[Contributions and outline](#)".

### Contributions and outline

This study aims to provide a resourceful, performance efficient, and low energy cost mobile cloud architecture. It facilitates a mobile client in executing resource-demanding applications. In the architecture, the resources allocated based on multi-criteria such as signal strength ( $-30$  dBm to  $-80$  dBm), application content (task size), and the communication barriers (signal quality and resource constraints). The performance evaluates executing resource-demanding applications using physical implementation. Results and analysis show performance efficiency, eradicates communication overheads, minimizes energy consumption, and task execution turnaround time. Hence, this architecture is a resourceful, performance efficient with low energy cost mobile cloud architecture for the resource-demanding applications.

The original contributions of this paper are:

- Architecture's motivation discusses in ("[Motivation](#)").
- Proposed architecture ("[Proposed architecture](#)") and signal strength analyzer (Section "[Signal strength analyzer](#)").
- Architecture service domain ("[Architecture's service domain](#)"), mobility (Section "[Mobility](#)"), resource-intensive (Section "[Resource intensive](#)"), and barriers in mobile communication (Section "[Barriers in mobile communication](#)").
- Proposed executable task estimation, offloading decision, and switching algorithms (Section "[Offloading decision algorithm](#)" and "[Switching decision algorithm](#)").
- The paper is completed by Section "[Introduction](#)", indicates the importance of efficient mobile cloud architecture for the resource-demanding applications and the purpose of the study. Section "[Related work](#)" presents the study's related work and the idea in the literature.

Section "[Experimental setup](#)" presents the experimental setup. Results and performance evaluation discusses in Section "[Results and discussions](#)".

Finally, we present the study's findings, conclusion, and future research direction.

## Related work

In recent times, the SMD users' expectations from their devices and preferences for their computational needs have changed beyond the intrinsic capabilities of SMDs. The SMDs' users expect an efficient resourceful execution environment for their mobile applications. MCC is a resourceful distributed computing architecture that augments SMDs' resources when the application in execution and highly scales resources via web-link. MCC architecture integrates different technologies, such as computing, communication, and Cloud technologies [4]. It leverages the service advantages such as utility computing [15], grid computing [16], virtualization, and assures service efficiency by the cloud service providers using service level agreements (SLAs). Mobile clients (SMDs' users) expect service that should be self-customized, reusable, highly portable [15], and low-cost models such as pay-on-consumption. Mobile clients intended to access MCC architecture for executing intensive applications in a resourceful environment. Partitioning and offloading are established schemes [1] that support intensive application execution and enhance performance efficiency by augmenting device resources.

## Code offloading

Computationally intensive code offloads from resource-constrained client's mobile device to remote cloud resources for computational efficiency and battery performance [14]. Basically, an application's user interface (UI) code executes in the device, and the rest of the code migrates to the cloud resources. A typical Android application creates a UI thread with a user interacts [5]; CloneCloud offers a similar scheme. Generally, many established offloading schemes offload computation-intensive codes based on the device intended augmentation (e.g., increasing battery efficiency) using the mobile cloud augmentation service-model. The offloading schemes include the decision-making process and based on computation augmentation techniques (CAT). The CAT provides both augmentation model such as (code offloading and service-oriented task delegation) and augmentation architecture such as (parallel execution and opportunistic mobile collaboration). Fig 1

The code offloading techniques enable SMDs to offload computation-intensive application's part to the resource-rich service-oriented cloud environment, leveraging the cloud infrastructure [2] for performance efficiency. In code offloading, the significant aspects are (1) identify the computation-intensive code and (2) partitioned the executable code, to offload. These are the significant concerns of application developers while developing programming models for mobile applications. Usually, code offloading techniques develop considering computation augmentation techniques using different programming models.

These techniques are based on the augmentation model and classified into four categories: *partitioned offloading*, *virtual machine (VM) migration*, *mobile agents-based offloading*, and *replication-based offloading* [14].

In portioned offloading, a computation-intensive code from a processing application determines and offloads to the intended cloud resource, such as a distant cloud. This offloading augments the device resources and improves performance efficiency while a client executes a resource-demanding application.

In the VM migration approach, the code offloading decision is taken, dynamically, during the code execution in VMs in the remote cloud. Based on the offloading requirements, a live VM migration considers. A live VM migration can be costly because of transferring VM image and the unstable network condition (e.g., communication barriers). Cloudlet is one of the examples of live VM; it discusses in the proposed framework [6] and also considers the latency issue and data access limit. The other offloading framework, CloneCloud, works at the MCC application layer discusses in [1], such as DalvikVM and Microsoft.NET.

In mobile agent-based offloading, the device agent requests application execution to its counterpart in the cloud. The cloud agent obtains the execution information and offloads the required code that needs to execute in the cloud. This offloading scheme possibly overcomes network connection instability since the device agent uses asynchronous communication to get cloud resources. Moreover, it is useful in task offloading when a method is called in the Java Agent Development Environment (JADE) [14]. Offloading code movability and running adaptively are the significant advantages offers by JADE. Whereas application multitancy, load balancing, and multi-users energy efficiency via wireless channel do not consider, and hence affect the Quality of Service (QoS). Most of these issues investigate and address by minimizing the number of transmission time slots on mobile devices, which depend on channel states [17]. However, the framework did not consider the device and the task transmission executions efficiency of the cloud.

Replication Based-offloading: almost all the offloading techniques based on the augmentation model require the decision process on application execution code offloading. The decision is indeed made on the intended benefits and offloads the code to the cloud resources. Indeed, such a decision takes for computation-intensive applications, and light code executes on devices. Network-intensive applications' execution still remains unsolved. A framework has proposed to address this issue and accelerating mobile application through flip-flop replication [18]. The study proposed an Android-based framework called Tango, and it automatically switches execution and output display

between a running application and its replica on remote resources and attempts to reduce application latency. The framework describes that only one replica handles the execution and ignores parallel execution. It delays execution time, overheads of excessive switching, and lack of access to native resources when non-deterministic events occur, such as keyboard input, voice input, camera input, sensors inputs, click of a window button, and more [5].

Our work considers unstable network conditions (i.e., frequency range and communication barriers), latency issues, and data access limits from VM migration. Application multitenancy, load balancing, and multi-users energy efficiency via wireless channels do not consider mobile agent-based offloading. Parallel execution, execution time, overheads of excessive switching, and lack of access to native resources, when non-deterministic events occur consider from replication based-offloading.

### Resourceful architecture

An architecture based on VM technology has developed to instantiate customized service software on a neighboring trusted environment swiftly, say Cloudlet, and accessible by nearby SMD's [6]. The architecture exclusively has designed for device mobility and called cloudlet-based, resource-rich, mobile computing. Some issues raised in the study, such as cloudlet providers' user enrichment policies and seamless assistance for mobile clients, need to be improved [6].

A similar inclusive MCC architectures approach has proposed [13]. It aims to reduce mobile device energy consumption and maximizes serviceability for mobile client users. It considers the multi-criteria decision making (MCDM) TOPSIS method for making application execution decisions. Other studies identify ultra-wideband (UWB), [19] and ZigBee network [20], which provide energy-efficient services for mobile clients and fail to describe service efficiency during application execution handoff among the resources [13].

A context-aware offloading algorithm considering context changes in heterogeneous mobile cloud resources discusses in [21]. It provides the utilization of MCC architecture's decision on wireless medium occurs at application runtime. It plans to create communication ability among cloud resources by handover strategy so the existing system can be more efficient. But, the study lefts user mobility models in the future investigation.

### Motivation

The aforementioned studies discuss MCC architecture rapid customization, customized services for emerging diverse mobile applications, seamless assistance for mobile clients, and context-aware offloading algorithms. Even, one of the studies [21] discusses the decision engine to handoff applications execution from one cloud

resource to another and another based on multi-criteria. These features are the motivation behind our work.

Our work aims a resourceful architecture, efficient service availability, and energy and performance efficiency; provides stable network-communication scenarios in MCC architectures for mobile clients. The proposed architecture exclusively has designed to improve client mobility, performance, and energy efficiency. This also has considered the rapid customization of MCC architecture for the emerging diverse resource-demanding applications.

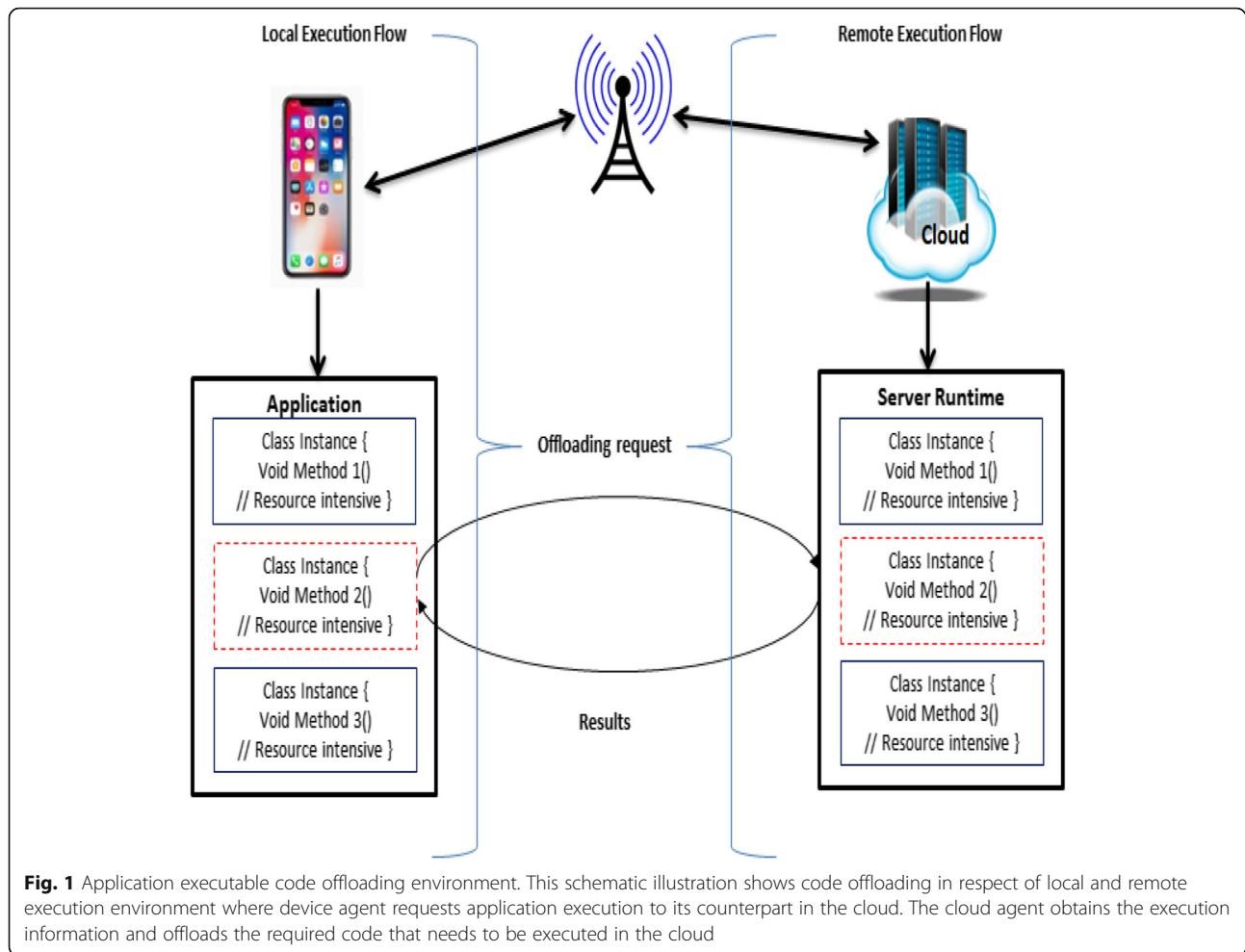
We propose a framework that switches application execution between cloud resources (collaborative cloud and distant cloud) via wireless channels (Bluetooth, Wi-Fi, 3G, etc.), [21]. In order to achieve both efficiencies service and energy, the framework considers potential obstacles: *mobility*, *intensive resources*, and *communication-barriers* (signal strength & quality, mobile traffic congestion, etc.), [22]. We proposed algorithms that help to achieve performance efficiency and minimizes energy consumption. We execute resource-demanding applications in the experimental setup. Despite our best efforts, we have not found the study's idea (resourceful architecture) that links in the literature. This study should be the first one that discusses the signal strength analyzer and the potential communication-barriers.

### Proposed architecture

Using MCC architecture, a mobile-client exploits cloud technologies to seamlessly execute its applications on cloud infrastructures via mobile Web-based services. With sustained effort by many researchers, several MCC architectures for mobile application execution proposed. Improving performance efficiency and minimizing device resource utilization are the most common purposes of these architectures. Some issues still are considered at different levels of MCC architectures to improve mobile application execution efficiency at minimum cost. Such as unstable network conditions (e.g., frequency range and communication barriers), latency, data access limit inability to access native resources, overheads of excessive switching, etc. [14].

We have designed an alternative resourceful MCC architecture (i.e., mRARSA = mobile resourceful architecture ready to serve applications) to improve performance efficiency and minimizes the utilization of device resources. Considering the issues which have mentioned in the previous paragraph and from several studies listed in [14]. In the literature, four mobile cloud physical layer architectures have discussed, (1) cloud computing with mobile terminals (distant cloud), (2) virtual cloud computing provider (mobile ad-hoc cloud), (3) Cloudlet, and (4) operator centric mobile cloud architecture (OCMCA), [4, 23]. Our designed framework exclusively is an integration of





the first three architectures. Moreover, Cisco plans to launch an innovative cloud-native architecture for mobile network ready for 5G<sup>1</sup>; it is entirely from radio access to the core for both network operations and services [24].

#### Architecture overview

Mobile hardware is resource-poor, whereas computation-intensive applications require both processing power and energy efficiency, which far outstrip mobile hardware's capabilities. The proposed architecture augments mobile device resources in a distributed mobile cloud environment. Figure 2 illustrates the architecture overview that leverages cloud resources switching between mobile ad-hoc cloud, cloudlet, and cloud. A mobile device accesses any of the three types of cloud resources via a mobile communication network [13].

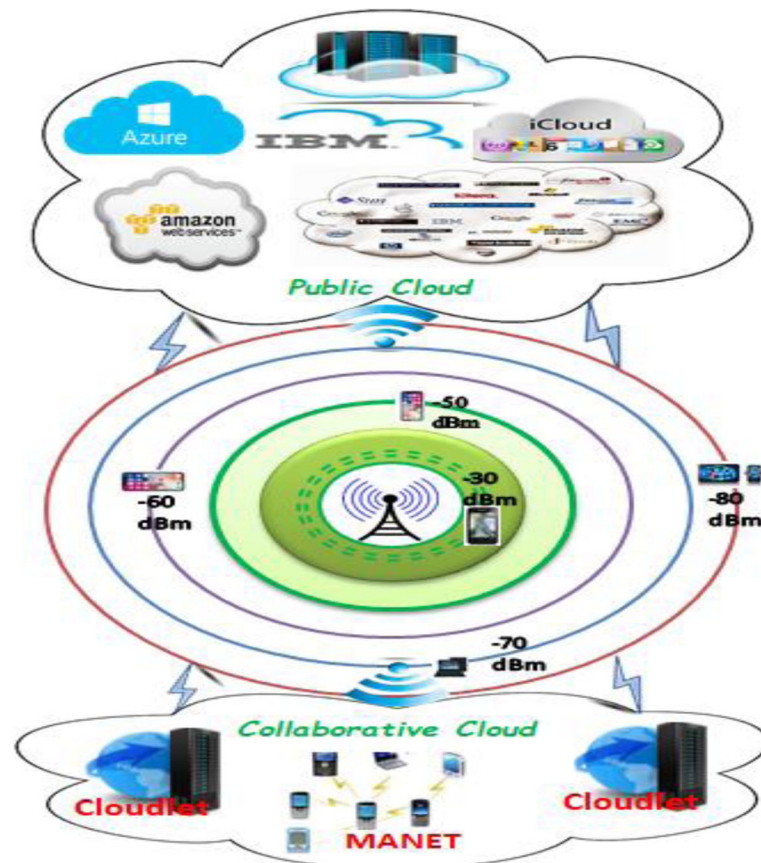
<sup>1</sup><https://www.prnewswire.com/news-releases/cisco-breaks-the-record-books-powering-rakutens-first-of-its-kind-cloud-native-mobile-network-300800833.html>

A mobile ad-hoc cloud is a virtual cloud and form by peer-to-peer mobile devices. The peer to peer (P2P) devices share resources if found in the vicinity of Wi-Fi range and interested in the contributing computing process. A mobile device able to access this cloud resource via short-range communication technologies such as Wi-Fi direct/ Bluetooth [21].

A cloudlet is a local resource that considers as multi-core processors and clusters in a box, where all significant computation occurs. A mobile client is allowed to access the cloudlet resource via Wi-Fi/WLAN [6].

A cloud is a most resourceful architecture wherein the subscribed mobile clients consume the computing service. The mobile clients connect to the Internet over a communication channel such as 4G/LTE to the remote cloud [4]. It is rich in availability, scalability, handover, native encryption, location privacy, and more.

This architecture consists of three core components: *a mobile device, a distant cloud, and a collaborative-cloud* (cloudlet + mobile ad-hoc cloud). In the collaborative-



**Fig. 2** Proposed architecture view. This architecture consists of three core components: a mobile device, distant cloud, and collaborative-cloud (cloudlet + mobile ad-hoc cloud). In the collaborative-cloud, a requesting mobile device able to share P2P nodes' computing resources based on computational requirements, and with respect to their Wi-Fi signal strength

cloud, a requesting mobile device able to share P2P nodes' computing resources based on computational requirements and concerning their Wi-Fi signal strength.<sup>2</sup> The requesting device based on the signal strength ( $-30$  dBm to  $-80$  dBm),<sup>3</sup> [25, 26] utilizes participating P2P resources based on the proposed switching algorithm decision; it switches either to collaborative-cloud or distant cloud. Further, it handoffs the executing application's tasks to more resourceful architecture, and utilizes the resources seamlessly. Moreover, the architecture remarkably supports mobile devices to switch between cloud resources based on signal strength analyzer i.e., Wi-Fi signal strength. The aspects mentioned above explain in subsequent sections.

<sup>2</sup><https://support.randomsolutions.nl/827069-Best-dBm-Values-for-Wifi>

<sup>3</sup><https://www.prnewswire.com/news-releases/cisco-breaks-the-record-books-powering-rakutens-first-of-its-kind-cloud-native-mobile-network-300800833.html>

### Essential components

This architecture's physical components distribute into three layers: *SMDs*, *collaborative-cloud*, and *remote cloud* (public cloud). However, the proposed framework components at application layer groups for mobile devices of the cloud resources. The mobile device components' group includes resource monitor, signal strength analyzer (SSA), decision module, task manager, communication manager, and failure recovery agent. Whereas, the cloud resource components group mainly includes the distant cloud, collaborative-cloud, task handler, and communication handler. Fig 3

### Resource monitor module

In this architecture, the resource monitor provides the system's resource information of dynamically profiling device resources such as the CPU, memory, battery, graphics, and network. It monitors and correlates different information such as battery consumption, IP traffic, location, and received signal strength [27]. These resources have a significant impact on device performance

**Table 1** Architecture comparison

Studies	MAHC	Cloudlet	Cloud	Integrated approach	Architecture criteria		
					Mobility	RI	CB
[6]	No	Yes	No	Yes	Yes	Yes	No
[8]	No	Yes	Yes	Virtualization	Yes	Yes	No
[9]	No	No	No	Distributed mobile setting	No	Yes	No
[10]	Yes	Yes	Yes	Service oriented based approach	Yes	Yes	No
[11]	No	No	Yes	Dalvik VM (Android)	No	No	No
[12]	No	No	Yes	Virtualization	No	Yes	No
[13]	Yes	Yes	Yes	MCDM	Yes	Yes	No
[14]	Yes	Yes	Yes	Six criteria	Yes	Yes	No
Our work	Yes	Yes	Yes	MCDM-TOPOSIS	Yes	Yes	Yes

MAHC Mobile ad-hoc cloud, RI Resource intensive, CB Communication barriers

during application computation. This resource monitor also assists the other modules, including the decision engine when requested, and majorly shares in achieving computational performance efficiency. Application developers consider profiling is one of the most critical tasks, and have to deal robustly.

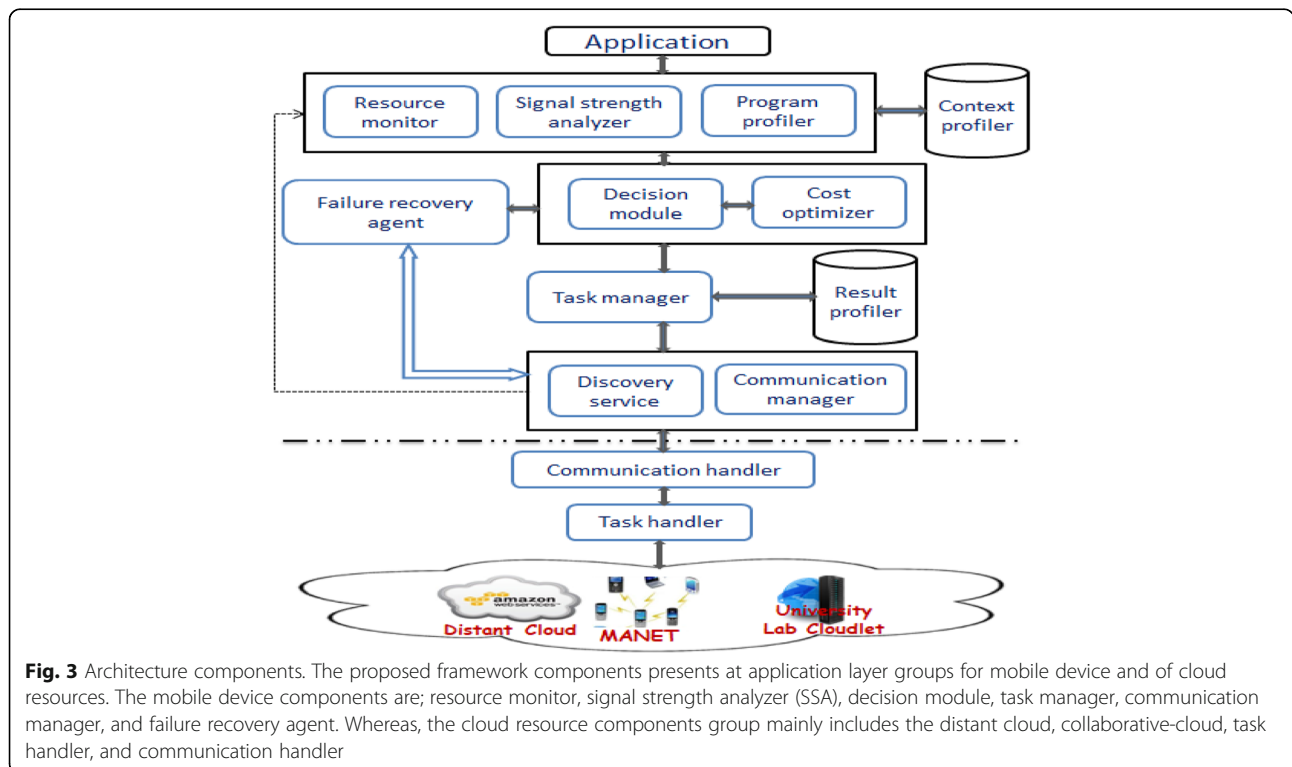
Generally, profilers categorize concerning hardware (device), software (application), and network (communication) [1].

Device profiler profiles the device’s physical components and represents the components operating conditions. These conditions considered by the decision engine during the cost estimation. Battery consumption

and the variant of CPU usage monitor to improve performance efficiency.

The program profiler measures the application parameters’ performances on the method level. It determines the types of cloud resources across this architecture and precisely executes the instructions. It also monitors the instructions execution time, methods’ call, and memory allocated for both instructions execution and the data input.

Network monitor asynchronously records the device network information at runtime during application execution. It monitors the network conditions such as the device-Wi-Fi connection states and bandwidth, device-



**Fig. 3** Architecture components. The proposed framework components presents at application layer groups for mobile device and of cloud resources. The mobile device components are; resource monitor, signal strength analyzer (SSA), decision module, task manager, communication manager, and failure recovery agent. Whereas, the cloud resource components group mainly includes the distant cloud, collaborative-cloud, task handler, and communication handler

Wi-Fi signal strengths, connection congestion level to utilize the cloud resources, and Bluetooth state. This information profile passes to the cost estimation model when requested for the network cost estimation.

#### **Signal strength analyzer**

The “*signal strength analyzer*” component is the novelty of the proposed architecture. This component included in the architecture minimizes the mobility issue while the client is executing the application in the physical proximity of the architecture. Notably, the mobility issue rises in a mobile ad-hoc cloud when the executable tasks distributed to different P2P contributing nodes. The different tasks of executing applications assign based on (1) Wi-Fi signal strength and (2) task content. The variants in signal strength (– 30 dBm to – 80 dBm) are reliable to execute different tasks shown in Table 3 of an executing application. The executable tasks dynamically assign to P2P nodes concerning the signal strength and the application content. The signal strength analyzer is one of the core components in the resource monitor module. It coordinates with the network monitor and communication manager.

#### **Decision module**

The decision module produces a significant impact on performance efficiency and determines the architecture resource (collaborative-cloud, or cloud) for the required services. Based on the running information, it decides the mobile task offloading strategies and dispatches the task’s package to the intended cloud resource. In this module, the decision engine and cost estimation agents are the most significant components. The cost estimation agent calculates the task execution time in eq. (7) that requires offloads to the intended cloud resources. It also calculates the device energy consumption in eq. (6) when the task is running. The decision engine necessarily provides tasks offloading decisions by executing the algorithm (5.1) and based on the updated information of coordinating modules. Further, the resource switching criteria, an application running information, and the two algorithms discussed in Table 5 and Section 5.1 and 5.2.

#### **Task manager**

The task manager collects intend task offloading information from the upper layer in the components’ hierarchy, and passes it to the communication manager. This information includes about offloading location in the network, method inputs, and the required libraries running the offloaded task. Then, the communication manager receives the information in a precise format and executes it by the offloading decision. Simultaneously, the task manager stores the task results in the

device database. Moreover, the task manager acts as a middle layer in the components hierarchy.

#### **Communication manager**

The communication manager is one of the core components of the proposed architecture. It handles the required connections for communicating its counter component (of remote cloud) across the architecture. It is a catalyst to the architecture performance efficiency and based on the effective coordination of communication handler and discovery service agent. Dynamically, it receives the decision engine offloading decision and executes the offloaded task. It also communicates the architecture’s failure, and then the recovery agent initiates recovery process service for any failure detection.

The discovery service agent determines architecture’s resources such as P2P nodes, collaborative-cloud, or the remote cloud for resource utilization. Further, the agent updates its information on the determined resource and then stores the updated information in the device profiler. The running resource utilization information, such as computation capacity, IP address, network congestion barrier, and signal strength, are crucial for the architecture service efficiency. The agent also communicates a signal strength analyzer to minimize resource searches such as P2P nodes in the architecture.

A communication handler is a component that communicates between the architecture’s resources and handles the resources running information. It provides the resource generated communication services of both client and cloud sides. It also handles the data and information, such as resource detection and location, failure detection, client and cloud synchronization states that have generated the executions. Further, when a mobile task offloads, then the communication handler wraps the related information (input, code, needed libraries) into the offloading package. As soon as the package is ready, it dispatches to the designated resource address for seamless processing. The resource location address (cloud server) determines by communicating the task manager. The communication handler of the cloud infrastructure unpacks the client package and synchronizes the missing libraries that execute on the server. Finally, after the execution completion, the result communicated to the client device. Effective communications between both the handlers contribute to performance efficiency.

#### **Failure recovery agent**

The proposed architecture precisely designed to share the resources and the running components information during task execution. In the architecture, the



failure recovery agent detects the failures (e.g., inaccessible node, signal barriers, and more) coordinating with other components communication manager and decision module. The agent dynamically detects the failures and instantly recovers the system execution.

### Architecture's service domain

In this architecture, the resources integrate to achieve service efficient application execution environment for mobile clients. The service framework inherits mobile cloud service-oriented characteristics shown in Table 2 and offers the resourceful services to the clients.

### Service metrics

This architecture considers some essential factors which significantly contribute to both performance efficiency and energy efficiency. Generally, these factors categorize as quantifiable (cost, energy-consumption, & communication-delay) and non-quantifiable (privacy, scalability, multicast, etc.), which represent subjective values [4]. These factors have to evaluate the desired quality services and performance efficiency. Usually, the cost is calculated for the executable mobile application, and of the device network usage. Often, the cost is low for mobile ad-hoc cloud, high for cloudlet, and low for the distant cloud [4] services, respectively.

Generally, client device energy-consumption measured in Joules (J), when executing applications, the lower energy consumption is desirable for service efficiency.

The communication-delay dynamically measures (in seconds-s) while executing applications in the architecture. During the execution, the communication parameters (processing delay, propagation, and transmission) fluctuate concerning cloud resources in the architecture. These fluctuations cause processing delays and have to measure for both in-house processing and offloading executable applications to the cloud destination.

The factors, as mentioned above, majorly contribute to performance efficiency, energy efficiency, and architecture's service performance.

### Service policies

The architecture dynamically provides only one of the cloud resources from (collaborative-cloud, or cloud) to the requesting mobile client. While allocating the resource, this architecture primarily considers three aspects: *mobility*, *resource-intensive*, and *signal barriers*, which influence performance efficiency. Essentially, the architecture also considers other criteria shown in Tables 4 and 5. Moreover, the architecture chooses the resources on the client's request and handles the potential obstacles which influence both performance and energy efficiency.

### Mobility

Mobility is one of the essential factors of this architecture and significantly considers seamless execution for mobile clients. Remarkably, this architecture improves mobility across the distributed mobile cloud architecture. Furthermore, it supports extended mobility without affecting the quality of running application execution, when covering intended distances before disconnecting from the mobile support station [28].

### Mobile ad-hoc cloud

This offers improved mobility, considers resource-intensive application requirements, and addresses signal barriers (i.e., signal strength) in the architecture. These factors improve client connection to resources, energy efficiency, and service availability. This architecture applies a one-hop Wi-Fi connection to all the participating P2P nodes. At present, the architecture considers only one hop Wi-Fi access point (AP) to determine the signal strength accuracy using the signal strength analyzer.

**Table 2** Architectures' characteristic matrix

Cloud architecture	Characteristics								
	AR	N	CE	M	Mob	US	O	SW	S
Mobil ad-hoc cloud	D	Mobile ad hoc network	Wi-Fi, Wi-Fi direct, Bluetooth	Self-managed	No	N specific users	Local	SOA	Only soft state
Cloudlet	D	WLAN latency/bandwidth	WLAN	Self-managed, little professional attention	Yes	Community user at runtime	Local & decentralized	SOA	Only soft state
Distant cloud	D	Internet latency / bandwidth	4G/LTE	Professionally administrator, service provider	Yes	N of users at all the time	Centralized & service centric by service provider	SOA	Hard & soft state
mRARSA	D	WLAN	Wi-Fi	Developer Management	Solve	Yes	Developer Ownership	SOA	Hard & Soft state

AR Architecture, N Network, CE Communication Environment, M Management, Mob Mobility, US User Support, O Ownership, SW Service framework, S State, D Distributed

**Table 3** Context vs. content tasks

Wi-Fi signal strength for executing applications' tasks			
Signal strength	Expected quality	Required for different tasks	Service efficiency
−30 dBm	Maximum - (nearest to access point)	Not typical or desirable in the real world.	Excellent
− 50 dBm	Anything down to this level can be considered excellent signal strength.	N/A	Good
−60 dBm	Good, reliable signal strength	Intensive task can be considered	Good
− 67 dBm	Minimum signal strength for applications that require very reliable, timely packet delivery.	VoIP/VoWiFi & non -HD video streaming	Good
− 69 dBm	Minimum signal strength for reliable packet delivery	Light browsing & email, web	Good
− 79 dBm	Minimum signal strength for basic connectivity. Packet delivery may be unreliable.	Connecting to the network	Fair
− 90 dBm	Approaching or drowning in the noise floor. Any functionality is highly unlikely.	Task hasn't defined yet	Average
(−90 to − 99) dBm	Functionality hasn't defined	Task hasn't defined yet	Below Average
− 100 dBm	N/A	Not acceptable	Poor
− 110 dBm	N/A	N/A	No signal

**Context aware P2P node connection** This architecture provides connections to other participating nodes in the range of Wi-Fi signal strength. The signal strength is measured in dBm (decibel milliwatts) and ranges in between (− 30 dBm to − 89 dBm) [25, 26]. This component signal strength analyzer analyzes the participating nodes' signal strength at different locations in the clients' signal strength range. It also determines both participating and non-participating nodes in the access point signal range.

**Application execution process** (1) the mobile client requests the job to other participating nodes in the signal range. The client job requester module dynamically divides (based on multi-criteria) the executable code into tasks, and distributes the tasks among participating nodes. The job requester sends the tasks, waits for the

processing, and receives the task-reply. Furthermore, the job requester receives the task reply, consolidates the reply, and then returns to the participating nodes. Further, the process repeats for all the executable tasks and completes the application execution. (2) Whereas the participating nodes receive the job requests (tasks), process the requested task, and returns the task's result to the client's job requester shown in Fig. 5. Moreover, the process repeats for each receives task and shares the nodes' resources in completing the client's execution requirements.

#### Client and nodes mobility

- This architecture provides mobility as one of the core characteristics during a client's job processing. It significantly improves mobility for both client devices and the participating nodes. The remarkable characteristic of this architecture is that the performance efficiency does not degrade in mobility during application processing. The resource allocation Fig. 5 shows the dotted lines either side (of the frequency range) represents the specific signal strength (e.g., − 30 to − 89 dBm). Here, mobility does not degrade the processing performances between dotted lines and towards access point and continues the sharing of resources between the devices seamlessly.

#### Cloudlet

It is the core component of collaborative-cloud. Usually, cloudlet servers install in thickly dense areas such as malls, chat areas, cafés, etc. collocated with Wi-Fi hotspots [4]. In the proposed architecture, a mobile client consumes cloudlet services as a thin

**Table 4** Offloading decision criteria

Energy consumption (EC)	Total turnaround time (TT)	Offloading decision
L	L	H
L	M	H
L	H	M
M	L	M
M	M	M
M	H	M
H	L	M
H	M	L
H	H	L

Low-L, Medium-M, High-H

**Table 5** Resource switching criteria

Energy consumption (EC)	Resource connection time (RCT)	Frequency range (FR)	Task workload (TS)	Switching decision
L	L	M	M, H	M
L, M	L	H	H	M
L	M	M	L	M
L, M	M	H	M, H	M
L	H	M	L, M	M
L	H	H	M, H	M
M	L	M	M	M
M	M	M	L, M	M
M	H	M, H	L, M	M
H	L	M, H	H	M
H	M	M	M, H	M
H	M, H	H	H	M
H	H	M	M	M
L, M, H	L, M, H	L	L, M, H	L
L	M	M	M, H	L
L, M, H	H	M	H	L
M	L, M	M	H	L
M	H	H	H	L
L, M, H	L	M	L	H
L	L	H	L, M	H
L, M, H	M	H	L	H
L, H	H	H	L	H
M, H	L	H	L, M	H
H	L	M	M	H
H	M	M	L	H
H	M	H	M	H
H	H	M	L	H
H	H	H	M	H

client via WLAN/Wi-Fi /Wi-Fi direct. This architecture provides one-hop, low latency, high-bandwidth, real-time interactive response to the nearby resources. The resources, physical proximity significantly contributes to the client's application execution performance. This performance gracefully degrades if the client moves away from the serving resources. Further, this leads to a fallback mode where three possible scenarios exist (1) continue in the distant cloud or the worst (2) resumes in the client device or (3) continue in collaborative-cloud resources in the architecture.

The client device profiler keeps track of the connection range profile records. Further, the profiler maintains neighboring resource availability to the

clients. It helps the signal strength analyzer in identifying the intended resource for the serving client. During the application execution, this architecture improves mobility with static resources. Currently, this architecture does not consider mobile cloudlets [29] for performance accuracy.

#### **Distant cloud**

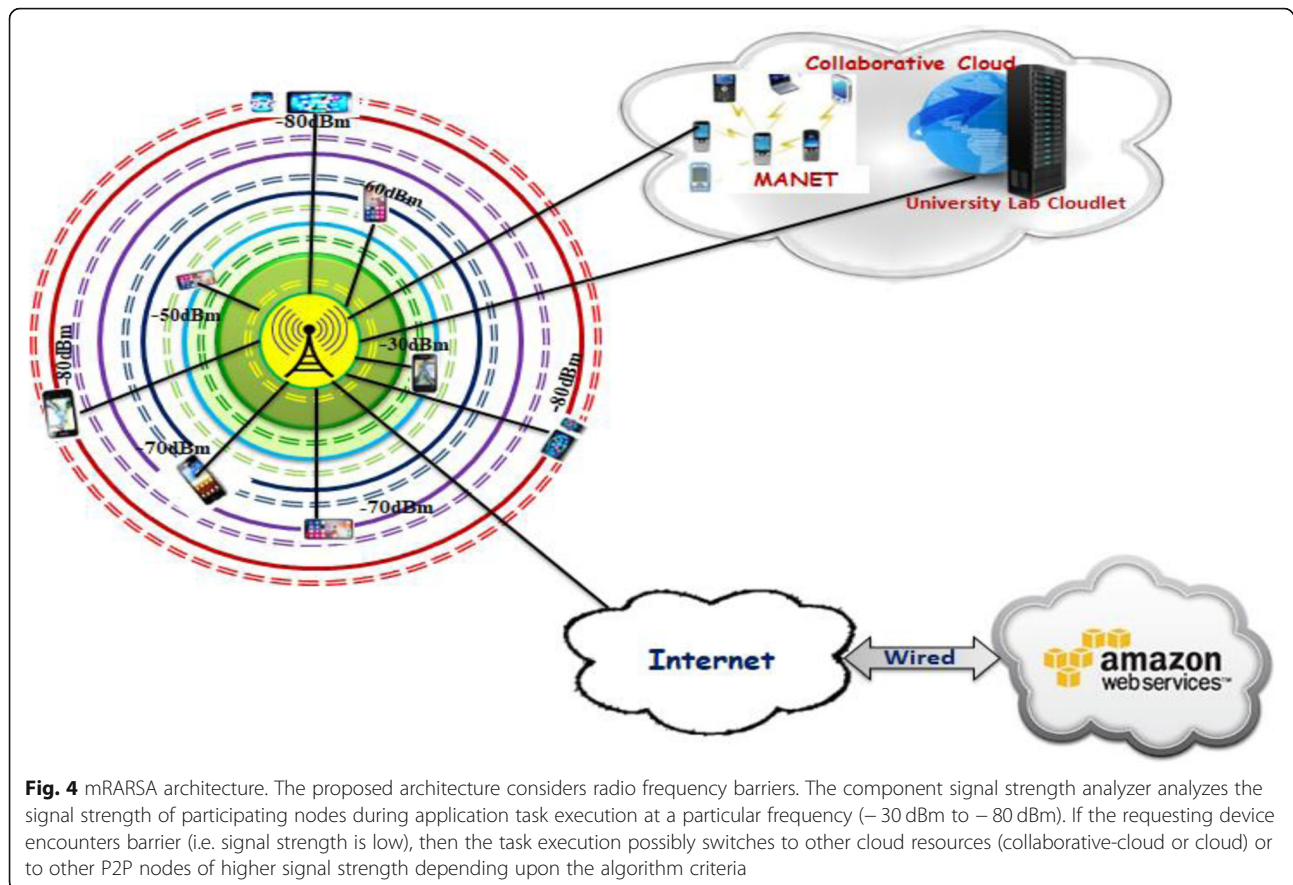
The proposed architecture inherits cloud computing mobility characteristics. During the execution of the application, the client avails mobility freedom in the entire area covered with subscribed service providers using network roaming. Mobile clients can also connect through Wi-Fi, but issues, such as network availability and handover, need to be considered when the client is nomadic.

#### **Resource intensive**

This architecture facilitates an integrated approach that significantly contributes to the execution of resource-demanding applications. During the execution, this architecture fairly improves energy and performance, the two most crucial mobile client concerns. It also dynamically decides the client's executable task-sharing resources and strategies (when, what, where, & how). These strategies correspond to the task's specific decision, task's intensive type, task's target location, and tasks' operations profile, respectively. Ideally, this architecture emphasizes energy and performance efficiency wherein mobile devices are resource constraints.

#### **Context vs. content aware resource allocation**

- The client's resource constraint device still needs resource augmentation during intensive application execution. This architecture's design introduces an application layer component, i.e., the signal strength analyzer that analyzes signal strength. It determines the P2P nodes, their locations, and the signal strength in the fair range of one access point Wi-Fi connection. Based on the signal strength, the client's application tasks assign to the participating nodes. It shows in Fig. 4 and describes in Table 3. The signal strength analyzer shown in Fig. 6 coordinates with the decision module, and the network monitor monitors nodes' Wi-Fi connection states and their bandwidths. The network monitor also monitors the signal strength congestion (communication barrier) level, and the barrier discusses in Section 4.5. Ideally, the client's task manager coordinates the communication manager, receives the decisions, and assigns intensive tasks



based on the node's signal strength (– 30 dBm to – 79 dBm).

#### **Collaborative and distant cloud**

This architecture consists of a collaborative and distant cloud for executing resource-intensive applications. Usually, the client device offloads the application's executable tasks to cloud resources based on different offloading criteria during the application processing. A recent study [30] discusses task offloading to (cloudlet or remote cloud) applying the mathematical concepts, stochastic geometry. This study analyzes the outage probability of task offloading between cloudlets and cloud in heterogeneous mobile cloud computing. Another study includes a network-aware offloading algorithm [31].

The proposed architecture applies context and content-aware task analyzing and task offloading algorithms. These algorithms apply the MCDM-TOPSIS model to select the potential cloud resource in the architecture under running context such as signal strength, task workload, data rate, etc. These factors significantly contribute to obtaining both performance and energy efficiency during intensive application execution. The algorithms discussed in Section 5.

#### **Barriers in mobile communication**

In the World Radio communication Conference organized by ITU Telecom World 2018,<sup>4</sup> discusses the growth in the usage of mobile communication systems by mobile users [32]. It emphasizes the effective usage of radiofrequency and spectrum management. Radiofrequency barriers are the major obstacles in the efficient usage of the frequency spectrum.

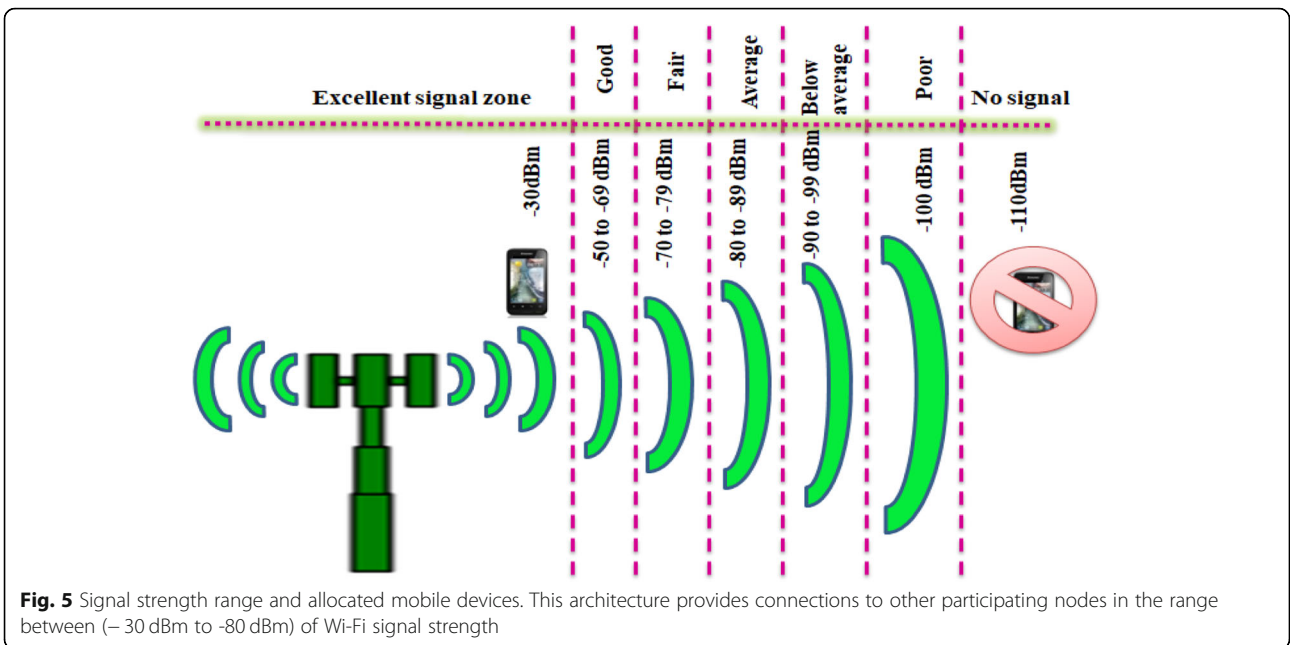
**Barriers** In mobile communication, the radio frequency spectrum that transmits signals from mobile devices to any other electronic receiver (e.g., TV,) whereas mobile devices use the lower frequency bands of the spectrum. The lower frequency bands can be blocked, reflected, or refracted by physical barriers<sup>5</sup> (buildings, trees, and rain), [33] and attenuated by frequency barriers (signal strength and quality, mobile traffic congestion), [22].

**Wi-fi performance efficiency** ideally, place wireless access point or router high up with a clear line of sight to,

<sup>4</sup>Itu-r, ITU-R Radio communication Study Groups D IO REGU L A. 2016

<sup>5</sup><https://www.ic.gc.ca/eic/site/069.nsf/eng/00088.html>





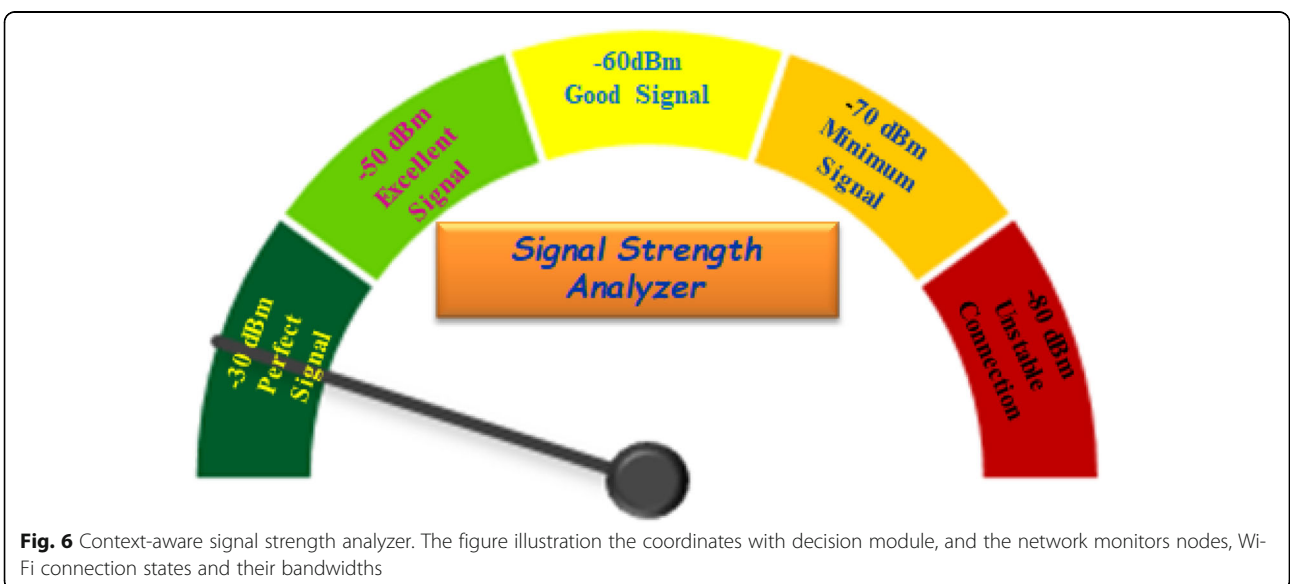
as considering large parts of the premises and performance efficiency. Protocol overhead declines signal strength and effects application throughput. The other factors, such as power output, receiver sensitivity, and path loss, have a direct impact and contribute to signal strength.

**mRARSA service design** The proposed architecture considers radiofrequency barriers. The component *signal strength analyzer* analyzes the signal strength of participating nodes during application task execution at a particular frequency say (- 60 dBm). If the

requesting device encounters a barrier (i.e., signal strength is low), then the task execution possibly switches (1) to other cloud resources (collaborative-cloud or cloud) or (2) to other P2P nodes of higher signal strength depending upon the criteria.

**Architecture algorithms**

This architecture focuses on the efficient utilization of cloud resources accessing through mobile devices via wireless mediums. The resource’s availability and performance efficiency based on multiple criteria such as signal strength, device context and application contents,



**Table 6** Preferred choice criteria

Data requirements	Computation requirements	Preferred choice
Low	Low	Mobile device
	Medium	Collaborative cloud
	High	Distant cloud
Medium	Low	Mobile device
	Medium	Collaborative cloud
	High	Distant cloud
High	Low	Collaborative cloud
	Medium	
	High	Distant cloud

network barriers, and energy consumption. This architecture includes two algorithms for (1) offloading decision (5.1) and (2) resource switching algorithm (5.2).

**Algorithms’ criteria**

This architecture considers offloading decisions based on Energy consumption (EC) and Total turnaround time (TT). The executable code offloads when the offloading task is high, and EC and TT are low. Similarly, the execution process continues, as shown in offloading decision criteria Table 4.

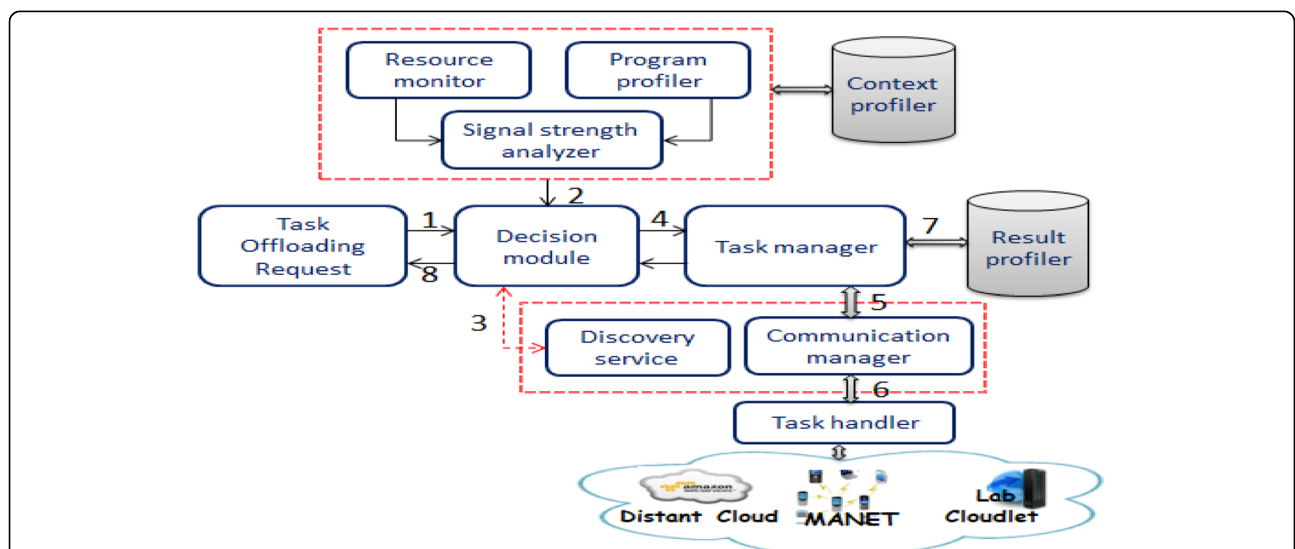
Table 4 includes two offloading criteria (EC and TT). The decision module determines the architecture resource based on the following priority: (i) *Offloading*

*decision high priority*, when the Frequency (FR) (30–49 dBm), and Task size (TS) (6-10 Mb) are in these limits. (ii) *Offloading decision medium priority*, when FR (50–69 dBm) and TS (3-6 Mb) are in these limits. (iii) *Offloading decision low priority*, when FR (70–89 dBm) and TS (0-3 Mb) are in these limits. Moreover, the cost estimation agent calculates the task execution total turnaround time and device EC to be offloaded to the intended cloud resources, and the offloading decision result depends on the algorithm 5.1.

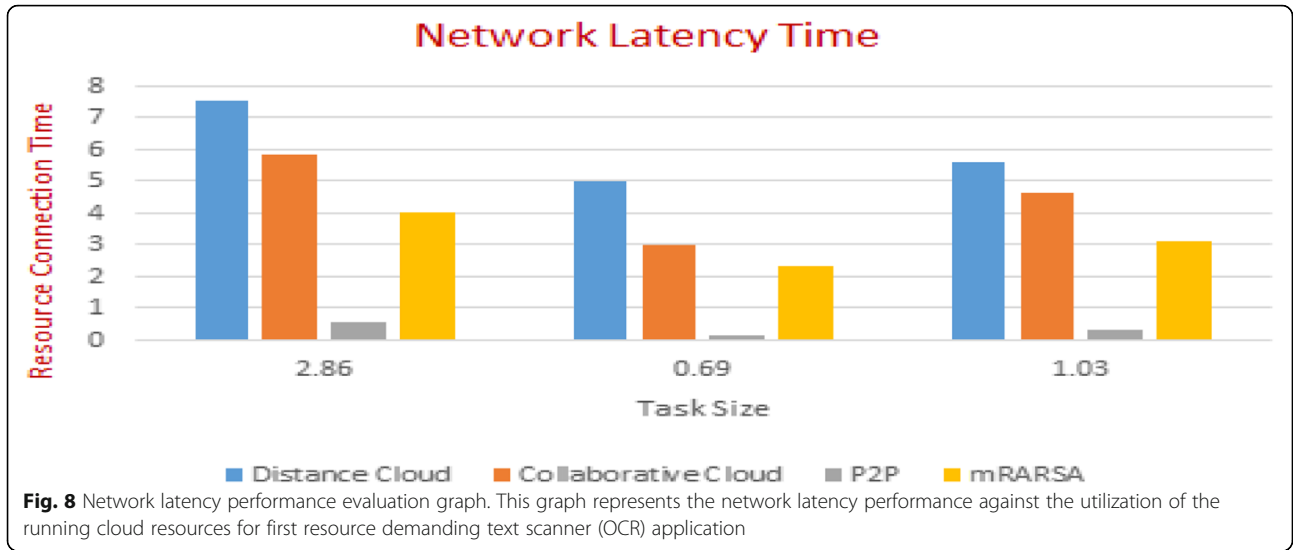
In Table 5, the resource switching criteria includes based on the four criteria (EC, RCT, FR, and TS). Here, the execution process (decision module) finds the intended resource in the architecture, and estimate connection time and device configuration. Based on these estimations and switching decision algorithm 5.2, the requesting device gets the preferred choice, as shown in Table 6.

**Offloading decision algorithm**

First, this algorithm applies to check the resources in the vicinity and determines the parameters, such as waiting time, communication energy, processing energy, and total connection time required to access the distant resources for the processing device. Here, these values are inputs to fuzzy-MCDM technique to choose the alternative cloud resources because the decision making is fuzzy in nature [34]. These processing parameters directly contribute to energy consumption and service



**Fig. 7** Task execution flow diagram. Figure 7 shows the task execution flow: 1. Sending task execution request to the decision module. 2. Collecting requested parameters from the resource monitor, program profiler, and signal strength analyzer. 3. Get the required information of the architecture resources. 4. Task manager begins communication once the task offloading decision is made. 5. Communication manager subtasks the job’s task for parallel processing. 6. The task handler offloads the task to the determined architecture resource. 7. Records parallel execution results, and store both the execution time and energy consumption in the result profiler. 8. Finally, the result sends back to the requesting mobile device



availability. Here, the executable task offloads when (i) the offloading criteria are high concerning the following parameters, waiting time ( $WG_{tot}$ ), transmitting time  $W_{trans}$ , processing time  $W_{proc}$  and (ii) receiving time ( $W_{rcv}$ ) is less. Likewise, the proposed algorithm calculates total turnaround time and the energy consumption for offloading the executable task.

Estimation model:

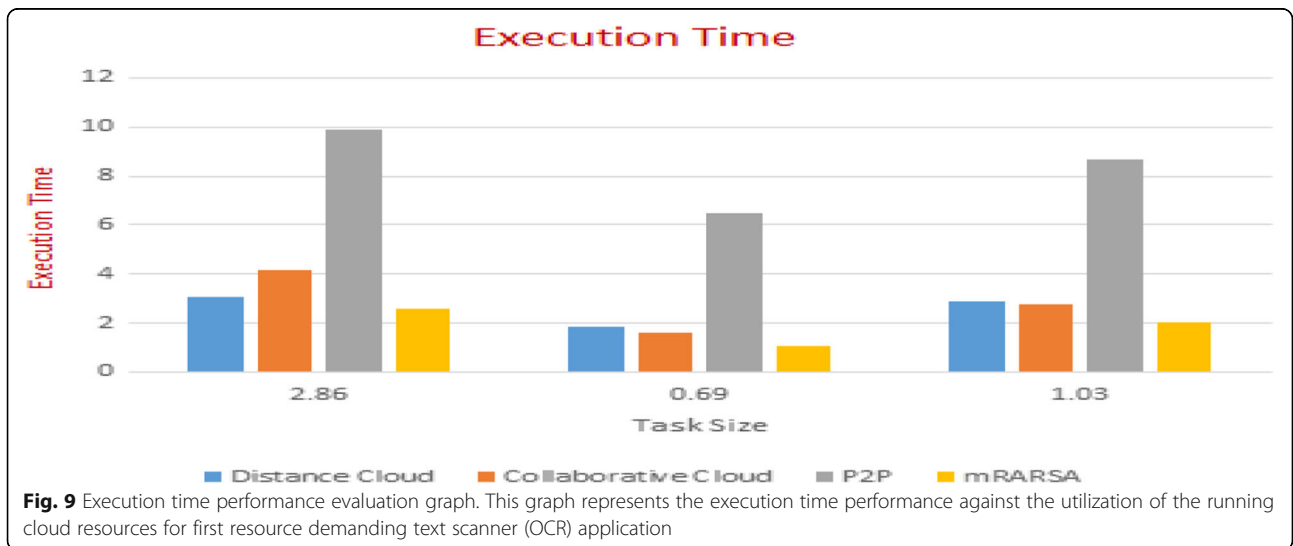
$$WG_{tot} = W_{trans} + W_{rcv} + W_{proc} \tag{1}$$

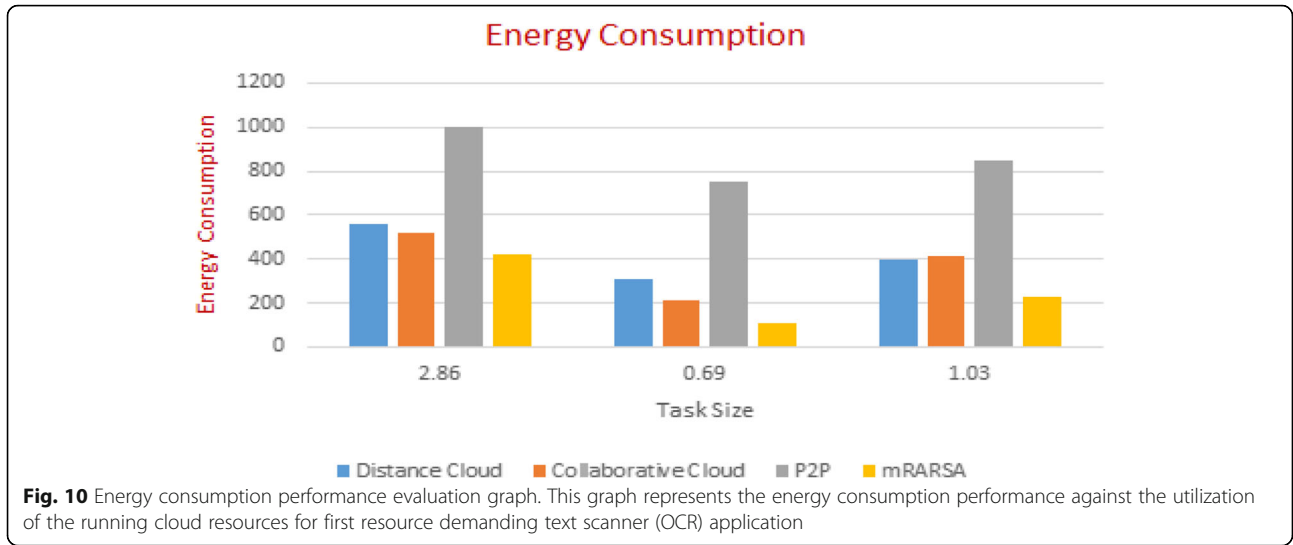
$$W_{trans} = \frac{S_{data}}{\alpha_d} \tag{2}$$

$$W_{rcv} = \frac{S_{data}}{\alpha_d} + \beta \tag{3}$$

$$W_{proc} = \frac{P_i \times CPPI}{PIPS} \tag{4}$$

In eq. (2), ( $S_{data}$ ) is the total data size that requires transmission, ( $\alpha_d$ ) is the data rate in the wireless medium, and  $\beta$  represents the frequency situations like throughput and data packet loss. Since it is calculated in communication error due to the loss of path, and its calculation does not consider here. Moreover, the processing client device owns the intelligence to determine communication frequency and estimates the value against  $\beta$ .





In eq. 4, ( $P_i$ ) is the number of instructions where  $i=0$  to  $n$  and  $n$  is the number of tasks, ( $CPPI$ ) is clock per instruction, and ( $PIPS$ ) is packet instruction per second. Further, ( $E_{tot}$ ) represents the total energy consumption in the task execution performance and ( $E_{mob}$ ) represents energy consumption in the mobile device in eqs. (5) and (6). In eq. (7), ( $T_{mob}$ ) represent the device processing capacity during the task execution.

$$E_{tot} = (E_{rt}) \times (W_{trans} + W_{rcv}) \times (W_{proc}) \tag{5}$$

$$E_{mob} = (PW_{mob}) \times (T_{mob}) \tag{6}$$

$$T_{mob} = \frac{Pi \times CPPI}{PIPS} \tag{7}$$

$$PW_{mob} = (V \times I) \tag{8}$$

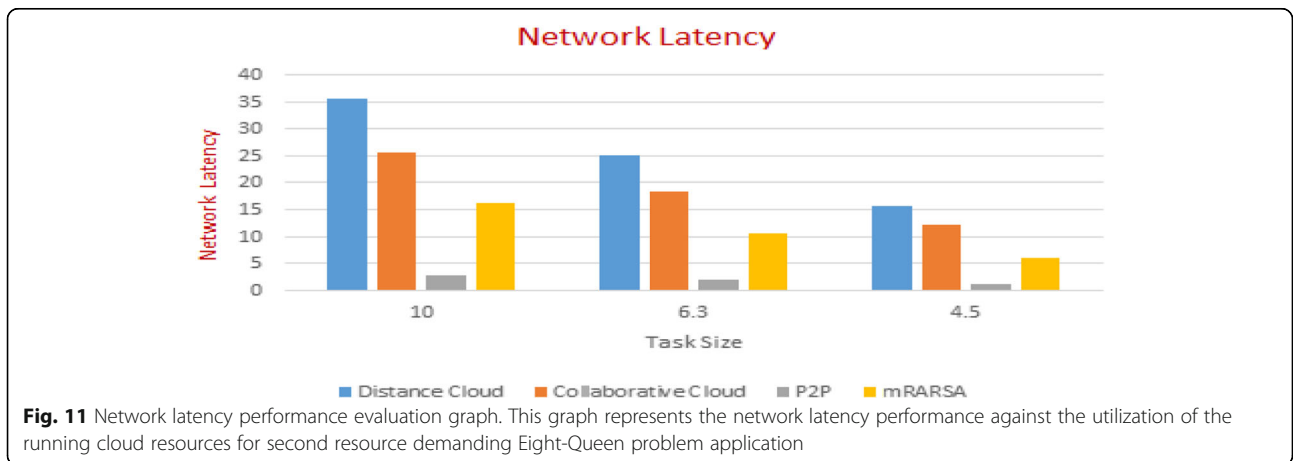
Where ( $E_{tot}$ ) is the total energy consumption in the processing device ( $E_{rt}$ ) is roundtrip energy consumption to transmit and receive data from a

processing device to cloud resources. ( $PW_{mob}$ ) represent power consumed in the client's mobile device, whereas  $V$  and  $I$  represent voltage and current, respectively.

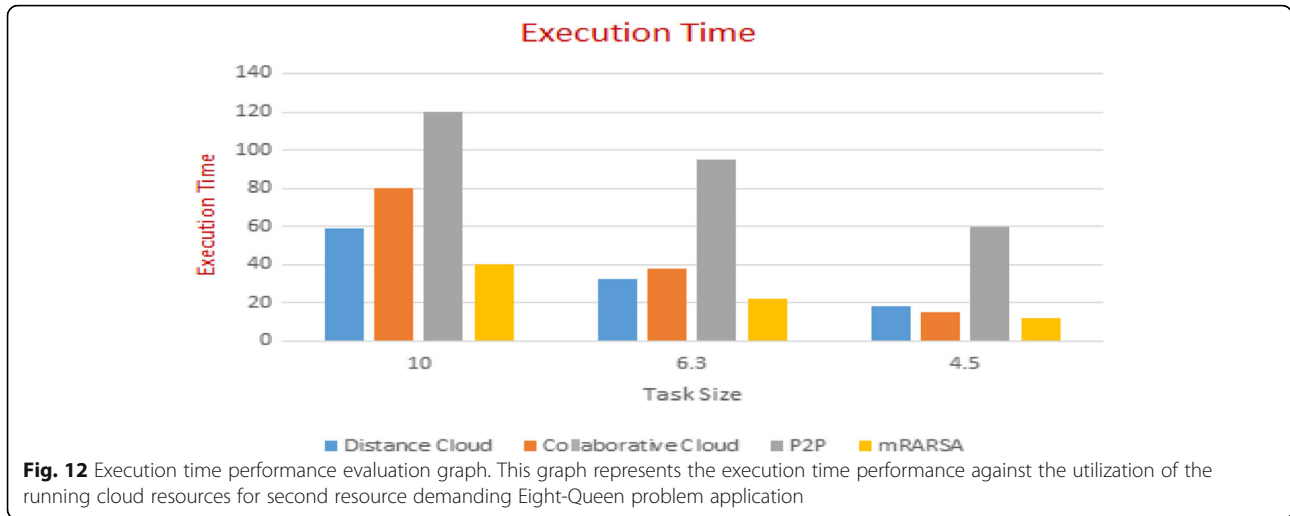
The connection time from the requesting device to the allocating resource measures by ( $T_{conn}$ ) shown in eq. (10). Where ( $F_{range}$ ) represents the frequency range and ( $D_{mobS}$ ) represents the distance between the device to the cloud resources. The connection time and the distance calculates based on the device mobility ( $M_{sp}$ ), clock time interval ( $CC_{rt}$ ), and ( $W_{sp}$ ) represents the standard wave speed. Further, the running distance calculates using eq. (11).

$$W_{sp} = 3 \times 10^8 m/s \tag{9}$$

$$T_{conn} = \frac{F_{range} - D_{mobS}}{W_{sp}} \tag{10}$$







$$D_{mobS} = (M_{sp}) \times (CC_{rt}) \tag{11}$$

**Switching decision algorithm**

This algorithm considers all the four criteria, as mentioned above, for making the tasks switching decisions.

```

(WLAN) AP ← All participating nodes are connected with single access point.
while (WLAN) AP ≠ & not null
  Consider, (CSWLAN) & (βWLAN) to search access node
  If (CSWLAN - βWLAN) > (Tthreshold) and (βWLAN > CCWLAN)
    TCcit = (N × (D / β(CCWLAN))) + (Tpd) + (Tid) + (Tcit)
    T(New)cit = (N × (D / β(CSWLAN))) + (Tpd) + (Tid) + (Tcit)
    If T(New)cit < T(Exi)cit
      CCWLAN = CSWLAN and β(CCWLAN) = β(CSWLAN)
      Then we assume in the network (Existing Net ID = Net id (CCWLAN))
    end If
    If T(New)cit > TRTT
      Switching trigger execute
    else
      Do not Switching trigger execute
    end If
    If switching is triggered
      Measure Total energy consumption (Etot), processing time (Wproc), total waiting time (WGtot) and connection time (Tconn).
      Execute MCDM-TOPSIS methodology based on resource switching criteria's
      Check idle peer mobile device to execute the task. If it is not suitable solution do not execute.
    else
      Switch following the criteria. /*Table 5 */
      Select the preferred choice. /*Table 6 */
      Process switching execution.
    end If
  end if
end while
    
```

Table 6 includes the resource obtaining criteria, data-intensive and, computation-intensive based on task requirements. The switching algorithm allocates the resource request based on the preferred choice criteria listed in the table.

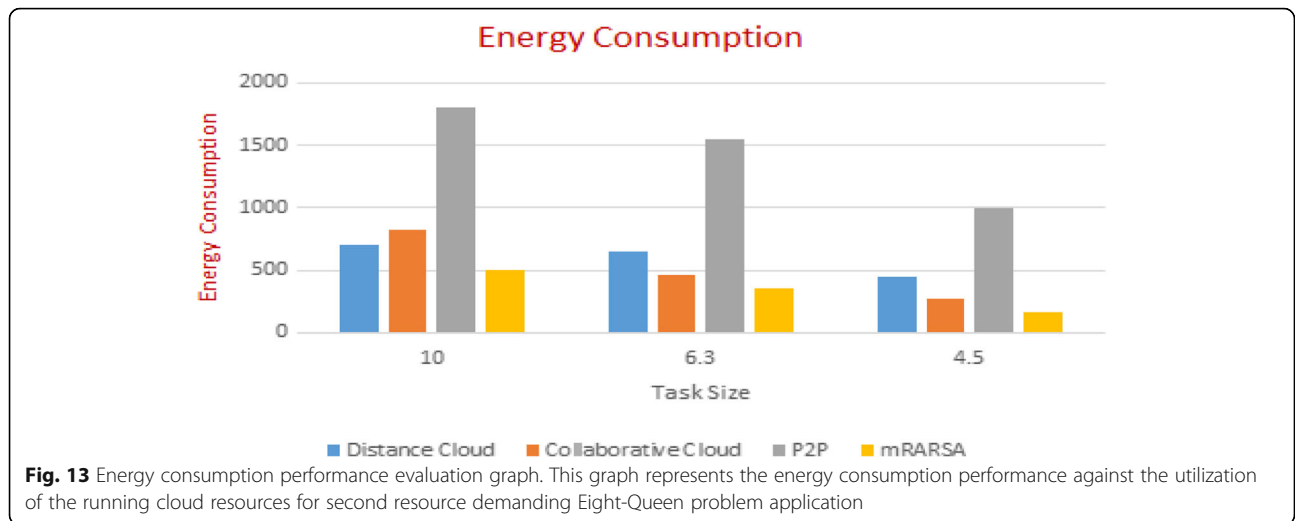
**Experimental setup**

We used three Lenovo K8 Note smartphone with the same configuration Deca-core (2.3 GHz, Dual core, Cortex A72 + 1.85 GHz, Quad-core, Cortex A53 + 1.4

GHz, Quad-core, Cortex A53), processor paired with 4 GB of RAM, 64 bit architecture, MediaTek MT6797D chipset, Mali-T880 MP4 graphics, and 4000 mAh Li-Polymer battery, Wi-Fi 802.11, b/g/n, v4.2 Bluetooth connectivity and Android v7.1.1 operating system. For the collaborative-cloud setup, we use Dell PowerEdge T320 server in the University computer lab, Saudi Arabia. The configuration such as Intel Xeon processor E5-1410 processor, total 12 GB RAM in (6 DIMM slots) with Microsoft Windows Server 2012 operating system. For the distant cloud, we used Amazon EC2 instance of Asia pacific (Mumbai) location with Microsoft Windows 2012 R2 Standard edition with 64-bit architecture, t3.xlarge type, and 16 GB RAM. Moreover, we have used PowerTutor 1.5<sup>6</sup> [35] to analyze the system’s performance and the application’s power usage. During the implementation, all unnecessary background services were terminated. Two android applications (OCR and Eight-Queen problem) of three different tasks’ sizes were implemented.

Figure 7 shows the task execution flow: 1. Sending task execution request to the decision module. 2. Collecting requested parameters from the resource monitor, program profiler, and signal strength analyzer. 3. Get the required information of the architecture resources. 4. Task manager begins communication once the task offloading decision is made. 5. Communication manager subtasks the job’s task for parallel processing. 6. The task handler offloads the task to the determined architecture resource. 7. Records parallel execution results, and store both the execution time and energy consumption in the result profiler. 8.

<sup>6</sup><https://www.ic.gc.ca/eic/site/069.nsf/eng/00088.html>



Finally, the result sends back to the requesting mobile device.

**Results and discussions**

The prototype testing includes two types of resource-demanding applications’ execution, one computation-intensive, and the other one is data-intensive. We evaluate the architecture’s performance against the utilization of running cloud resources. Essentially, performance evaluation considers network latency, execution time, and total energy consumption; these have explained through the graphs in figures (Figs. 8, 9, 10, 11, 12 and 13). Whereas, the client’s device battery power consumption is measured by PowerTutor 1.5. Further, the execution performance discusses in two scenarios.

**Scenario 1 text scanner (OCR) application execution**

The nature of OCR-text scanner application is to recognize any text from an image with (98–100) % accuracy and support more than 50 languages. This application is widely used in the Arab World to translate Arabic text into English. The application’s execution performance explains in Tables 7, 8 and 9). And the performance measure also explains through graphs, shown in Figs. 8, 9 and 10).

**Table 7** Network latency performance evaluation for OCR

	Task size (Mb)	Distant cloud	Collaborative-cloud		mRARSA
			Lab cloud	P2P	
Network latency (s)	2.86	7.52	5.86	0.55	4.05
	0.69	4.99	3.01	0.14	2.32
	1.03	5.62	4.62	0.32	3.14

Table 7 shows the three task sizes of the application. For distant cloud and collaborative-cloud, the latency time is higher. For mRARSA, the latency time is lesser than the other two. Indeed, the P2P latency time is much lesser among all, because the participating devices are in close physical proximity and access through one hop. The graphical representation shows through graph Fig. 8.

Note: Similarly, Tables 8 and 9 show the performances of execution time and energy consumption in the client device, respectively. Whereas, Figs. 9 and 10 describe the graphical representation of both tables

**Execution analysis (scenario 1)**

- For network latency, mRARSA consumes less time and resources for the three executed task sizes next to P2P.
- For execution time, mRARSA performs better than the other two architecture’s clouds.
- For energy efficiency, P2P consumes more energy, and mRARSA consumes less compare to the distant cloud.

**Scenario-2 eight-queen problem application execution**

This problem identifies eight positions to place the queens on an 8 × 8 chessboard. A robust computation

**Table 8** Execution time performance evaluation

	Task size (Mb)	Distant cloud	Collaborative-cloud		mRARSA
			Lab cloud	P2P	
Execution Time (s)	2.86	3.05	4.16	9.91	2.55
	0.69	1.82	1.6	6.48	1.05
	1.03	2.86	2.75	8.67	2.02

**Table 9** Energy consumption performance evaluation

	Task size (Mb)	Distant cloud	Collaborative-cloud		mRARSA
			Lab cloud	P2P	
Energy consumption (mA)	2.86	560	520	1000	420
	0.69	310	210	750	110
	1.03	400	410	850	230

is required to move between the positions. We have executed three different tasks' sizes applications namely (8-Regina 4.5 Mb, the queens 6.3 Mb and 8 Queens 10 Mb). The performance execution describes in Tables 10, 11 and 12), and the graphical representation through performance graphs, shown in Figs. 11, 12 and 13).

Table 10 includes three task sizes of the application. The nature of the application involves a high number of iterations during the task executions. Indeed, the latency time should also be high for the three sizes. Here, mRARSA takes lesser time than the other two distant and collaborative-clouds. The physical proximity of the participating devices improves P2P latency performance and takes lesser time compared to all.

Figure 11 shows the network latency performance inline to the four architectures' values of Table 10. It shows the graphical representation of network latency performance evaluation consumed by the four architectures against the task sizes.

Similarly, execution time Table 11 and energy consumption Table 12 has included the performance evaluation for the client device.

Figure 12 represents the execution time performance evaluation of the four architectures against three execution task sizes (10, 6.3, and 4.5) in Mb.

Figure 13 shows the energy consumption performance evaluation of the four architectures.

### Execution analysis (scenario 2)

- The latency performance evaluation for three task sizes (10, 6.3, and 4.5) in Mb shows the architecture performances in descending order

**Table 10** Network latency performance evaluation for Eight-Queen problem

	Task size (Mb)	Distant cloud	Collaborative-cloud		mRARSA
			Lab cloud	P2P	
Network latency (s)	10	35.52	25.62	2.75	16.21
	6.3	24.99	18.22	1.84	10.48
	4.5	15.62	12.08	1.04	6.02

(i.e., distance cloud, collaborative-cloud, and mRARSA).

- For the execution time evaluation, P2P takes the highest time. Distance and collaborative-cloud consume almost the same average time of three sizes. Whereas the performance of mRARSA much better during the tasks' execution.
- For energy consumption, P2P consumes the highest power in ascending order with respect to task sizes, due to devices are resource constraint in nature. Here, the distance cloud consumes high energy if the task size is < 10 Mb and low for > 10 Mb, since it is rich in resources. Collaborative-cloud consumption is just opposite to distance cloud. mRARSA is the comparatively better option for <= 10 Mb task size due to the proposed switching algorithm.

### Challenges and future work

The integration of cloud computing architectures with mobile communication for energy-efficient services is always challenging: Leveraging the architecture's resources efficiently during resource-demanding applications execution on a thin-client. Wireless communication latencies constitute a significant concern that delays both (1) allocation of needed resources (2) crispness of system response. Optimizing battery life during resource-demanding application execution is critical in achieving performance efficiency. Despite adequate bandwidth, the application execution throughput suffers due to a high degree of latency and round-trip time network latency. The architecture collaborative cloud's physical proximity impacts application execution performance. This performance gracefully degrades, if no resource is available nearby.

**Table 11** Execution time performance evaluation

	Task size (Mb)	Distant cloud	Collaborative-cloud		mRARSA
			Lab cloud	P2P	
Execution Time (s)	10	3.05	4.16	9.91	2.55
	6.3	1.82	1.6	6.48	1.05
	4.5	2.86	2.75	8.67	2.02

**Table 12** Energy consumption performance evaluation

	Task size (Mb)	Distant cloud	Collaborative-cloud		mRARSA
			Lab cloud	P2P	
Energy consumption (mA)	10	560	520	1000	420
	6.3	310	210	750	110
	4.5	400	410	850	230

Necessarily, this collaborative-cloud would be self-managing and required to be grouped into clusters of computing power, protected in the boxes. An ideal collaborative-cloud should support any resource-demanding application while executing with minimal software constraints.

### Deployment challenges

Several mobiles cloud architectural characteristics must be considered before designing the proposed architecture. Frequency analyzer and signal strength determiner are the major components that contribute to both service and energy efficiency. For effective performance, several criteria must be considered, such as interface availability, channel link speed, energy cost, mobile data cost (on use), signal strength, and communication barriers.

### Conclusion

This is a resourceful architecture that optimizes the cloud resources for a mobile thin-client. It considers both content (tasks' data) and context (signal strength and communication barriers) when allocating the resources to a requesting mobile client. It minimizes the communication overhead between the device and the architecture resources (P2P nodes, collaborative cloud, and cloud). It also leverages the architecture resources for the execution of resource-intensive mobile client applications. Essentially, this study proposes an algorithm that deals with both context-aware and content-aware decisions. These decisions occur at run time while a mobile client is executing applications to leverage the architecture's resources. These resources allocate via a one-hop access point through wireless communication. The deployment model supports any types of resource-intensive applications. The signal strength analyzer is a remarkable feature that determines signal strength and forwards the execution request based on content and tasks' size. The application classifies into executable tasks of different sizes. Then, these tasks are assigned to resources (collaborative-cloud and distant cloud) based on the signal strength and the task content. The execution model shows efficient results and provides quality, efficient decisions based on the current context. The client device achieves lower energy cost and execution time.

For execution, mRARSA is the best parameter architecture for execution and energy efficiency. We define the applications based on the following criteria: For a new process, estimate the data size, computation, and energy requirements (data, computation, and energy requirements: low, medium & high).

In our experiments, we implemented the resource-demanding applications in two categories: data-intensive and computation-intensive with different application sizes with the range between 0.69 to 2.86 Mb and 4.5 to 10 Mb, and their results are shown graphically.

We plan to execute different application sizes (i.e. > 10 Mb) at each signal strength (e.g. -67 dBm) and monitors client mobility. The task's execution contributes to another participating P2P node. This task switches to another resource to minimize failure and efficiently avoid barriers. The intended future work would monitor the expected barriers which affect both service efficiency and energy efficiency. More effective strategies should be developed to minimize the barrier effects and create the ability to inter-communicate cloud resources.

Optimistically, this architecture should provide more mobility with determining signal strength at the different range and switching task execution among participating P2P nodes of the architecture. This has to support the mobile clients' mobility in designated physical proximity of this architecture. Therefore, this should be a remarkable architectural feature for the aspirant researchers that need to be investigated in the future.

### Abbreviations

AP: Access point; CAT: Computation augmentation techniques; CB: Communication barriers; CPPI: Clock per instruction; dBm: Decibel milliwatts; EC: Energy consumption; FR: Frequency range; JADE: Java agent development environment; MAHC: Mobile ad hoc cloud; MCC: Mobile cloud computing; MCDM: Multi-criteria decision making; mRARSA: Mobile resourceful architecture ready to serve applications; OCMCA: Operator centric mobile cloud architecture; P2P: Peer to peer; PIPS: Packet instruction per second; QoS: Quality of service; RCT: Resource connection time; RI: Resource intensive; SMD: Smart mobile device; SSA: Signal strength analyzer; TT: Turnaround time; UI: User interface; UWB: Ultra-wideband; VM: Virtual machine

### Acknowledgements

The authors would like to express their gratitude to King Khalid University, Abha, Saudi Arabia for providing administrative and technical support such as computer lab equipment.



### Authors' contributions

AI (corresponding author): His contribution includes conceptualization, methodology design; manage required software, execute experiments and analyze results. He conducted a research and investigation process. Importantly, he owned responsibilities for the research activity, planning, and execution. AK: His contribution includes oversight and leadership responsibility for the research activity planning and execution, including mentorship external to the core team. KM: Contribution for this study includes conceptualization, investigation, methodology, supervision, validation, visualization, writing-original draft, and writing-review & editing. SY: Her contribution includes providing resources such as study materials, reagents, materials, computing resources. Specifically, critical review, commentary, and revision including prepublication stages are included. MAK: He is officially holding responsibilities of computer labs in the college. His contribution includes setting and configuring of cloud labs. MRH: He has contributed in editing and improving diagrams 'quality. All authors read and approved the final manuscript.

### Funding

The authors extend their appreciation to the Deanship of Scientific Research at King Khalid University for funding this work through Research Group Project under grant number R.G.P. 1/166/40.

### Availability of data and materials

Not applicable.

### Competing interests

This manuscript has not been submitted in any another journal or other publishing venue.

### Author details

<sup>1</sup>Banasthali Vidyapith, Niwai, Rajasthan, India. <sup>2</sup>King Khalid University, Abha, Saudi Arabia. <sup>3</sup>Suresh Giyan Vihar University, Jaipur, Rajasthan, India.

Received: 28 September 2019 Accepted: 19 January 2020

Published online: 07 February 2020

### References

- Gu F, Niu J, Qi Z, Atiquzzaman M (2018) Partitioning and offloading in smart mobile devices for mobile cloud computing: state of the art and future directions. *J Netw Comput Appl* 119:83–96. <https://doi.org/10.1016/j.jnca.2018.06.009>
- Wu H, Sun Y, Wolter K (2018) Energy-efficient decision making for Mobile cloud offloading. *IEEE Transactions on Cloud Computing* 7:161. <https://doi.org/10.1109/TCC.2018.2789446>
- Shaukat U, Ahmed E, Anwar Z, Xia F (2016) Cloudlet deployment in local wireless networks: motivation, architectures, applications, and open challenges. *J Netw Comput Appl* 62:18–40. <https://doi.org/10.1016/j.jnca.2015.11.009>
- Bou Abdo J, Demerjian J (2017) Evaluation of mobile cloud architectures. *Pervasive and Mobile Computing* 39:284–303. <https://doi.org/10.1016/j.pmcj.2016.12.003>
- Hung SH, Shih CS, Shieh JP et al (2012) Executing mobile applications on the cloud: framework and issues. *Comput Math Appl* 63:573–587. <https://doi.org/10.1016/j.camwa.2011.10.044>
- Satyanarayanan M, Bahl P, Caceres R, Davies N (2009) The case for VM-based cloudlets in Mobile computing. *IEEE Pervasive Computing* 8:14–23. <https://doi.org/10.1109/MPRV.2009.82>
- Bhattacharya A, De P (2017) A survey of adaptation techniques in computation offloading. *J Netw Comput Appl* 78:97–115. <https://doi.org/10.1016/j.jnca.2016.10.023>
- Cuervo E, Balasubramanian A, Cho D-K, et al (2010) MAUI: Making Smartphones Last Longer with Code Offload
- Dou A, Kalogeraki V, Gunopulos D et al (2010) Misco. 1. <https://doi.org/10.1145/1839294.1839332>
- Akan O, Bellavista P, Cao J, et al (2010) Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 76 Editorial Board
- Chun B-G, Ihm S, Maniatis P, et al (2011) CloneCloud: Elastic Execution between Mobile Device and Cloud
- Kosta S, Aucinas A, Hui P et al (2012) ThinkAir: dynamic resource allocation and parallel execution in the cloud for mobile code offloading. *Proceedings - IEEE INFOCOM*, pp 945–953. <https://doi.org/10.1109/INFOCOM.2012.6195845>
- Ravi A, Peddoju SK (2015) Handoff strategy for improving energy efficiency and cloud service availability for Mobile devices. *Wirel Pers Commun* 81: 101–132. <https://doi.org/10.1007/s11277-014-2119-y>
- Zhou B, Buyya R (2018) Augmentation techniques for Mobile cloud computing. *ACM Comput Surv* 51:1–38. <https://doi.org/10.1145/3152397>
- Wei Y, Blake MB (2010) Service-oriented computing and cloud computing: challenges and opportunities. *IEEE Internet Comput* 14:72–75. <https://doi.org/10.1109/MIC.2010.147>
- Foster I, Zhao Y, Raicu I, Lu S (2008) Cloud computing and grid computing 360-degree compared. *Grid Computing Environments Workshop, GCE 2008*, pp 1–10. <https://doi.org/10.1109/GCE.2008.4738445>
- Liu X, Yuan C, Yang Z, Zhang Z (2016) Mobile-agent-based energy-efficient scheduling with dynamic channel acquisition in mobile cloud computing. *J Syst Eng Electron* 27:712–720. <https://doi.org/10.1109/JSEE.2016.00074>
- Gordon MS, Hong DK, Chen PM et al (2015) Accelerating Mobile applications through Flip-flop replication. *Proceedings of the 13th annual international conference on Mobile systems, applications, and services. MobiSys 15:137–150*. <https://doi.org/10.1145/2742647.2742649>
- Merz R, Widmer J, Le Boudec JY, Radunović B (2005) A joint PHY/MAC architecture for low-radiated power TH-UWB wireless ad hoc networks. *Wirel Commun Mob Comput* 5:567–580. <https://doi.org/10.1002/wcm.313>
- Nadimi ES, Jørgensen RN, Blanes-Vidal V, Christensen S (2012) Monitoring and classifying animal behavior using ZigBee-based mobile ad hoc wireless sensor networks and artificial neural networks. *Comput Electron Agric* 82: 44–54. <https://doi.org/10.1016/j.compag.2011.12.008>
- Dastjerdi AV, Zhou B, Buyya R et al (2015) mCloud: a context-aware offloading framework for heterogeneous Mobile cloud. *IEEE Trans Serv Comput* 10:797–810. <https://doi.org/10.1109/tsc.2015.2511002>
- Mohiuddin K, Mohammad AR, Raja AS, Begum SF (2012) Mobile-CLOUD-mobile: is shifting of load intelligently possible when barriers encounter? *Information science and digital content technology (ICIDT), 2012 8th international conference on*, pp 326–332
- Fernando N, Loke SW, Rahayu W (2013) Mobile cloud computing: a survey. *Futur Gener Comput Syst* 29:84–106. <https://doi.org/10.1016/j.future.2012.05.023>
- The Network CTNS (2019) Cisco breaks the record books: powering Rakuten's cloud native Mobile network | the network. In: *The Network, Cisco's Technology News Site* [eyesaa.com/wi-fi-signal-strength/](https://www.eyesaa.com/wi-fi-signal-strength/) No Title
- LiveAgent (2016) Best dBm values for Wifi. In: *LiveAgent*
- Díaz A, Merino Gomez P, Rivas Tocado F (2010) Mobile application profiling for connected mobile devices. *IEEE Pervasive Computing* 9:54–61. <https://doi.org/10.1109/MPRV.2009.63>
- Qi H, Gani A (2012) Research on mobile cloud computing: review, trend and perspectives. *2012 2nd international conference on digital information and communication technology and its applications, DICTAP 2012*, pp 195–202. <https://doi.org/10.1109/DICTAP.2012.6215350>
- Panigrahi CR, Sarkar JL, Pati B (2018) Transmission in mobile cloudlet systems with intermittent connectivity in emergency areas. *Digit Commun Netw* 4:69–75. <https://doi.org/10.1016/j.dcan.2017.09.006>
- Lee H-S, Lee J-W (2018) Task offloading in heterogeneous Mobile cloud computing: modeling, analysis, and cloudlet deployment. *IEEE Access* 6: 14908–14925. <https://doi.org/10.1109/ACCESS.2018.2812144>
- Sarathchandra Magurawalage CM, Yang K, Hu L, Zhang J (2014) Energy-efficient and network-aware offloading algorithm for mobile cloud computing. *Comput Netw* 74:22–33. <https://doi.org/10.1016/j.comnet.2014.06.020>
- ITU-R (2016) ITU-R Radiocommunications Study Groups D IO REGU L A
- Communications Research Center (CRC) C (2018) Breaking the frequency barrier: using millimetre waves for Mobile services - communications research Centre Canada. In: *Government of Canada*
- Balioti V, Tzimopoulos C, Evangelides C (2018) Multi-criteria decision making using TOPSIS method under fuzzy environment. *Application in Spillway Selection Proceedings*, vol 2, p 637. <https://doi.org/10.3390/proceedings2110637>
- PowerTutor: A Power Monitor for Android-Based Mobile Platforms. Retrieved from <http://ziyang.eecs.umich.edu/projects/powertutor/>

### Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.