Optimizing Mobile Computing -Offloading through Data Size-Based Algorithm (DSBA)

MSc Research Project
Cloud Computing

Murugalakshmi
Student ID: x21207232

School of Computing
National College of Ireland

Supervisor: Yasantha Samarawickrama

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Murugalakshmi
Student ID:	x21207232
Programme:	Cloud Computing
Year:	2023
Module:	MSc Research Project
Supervisor:	Yasantha Samarawickrama
Submission Due Date:	14/08/2023
Project Title:	Optimizing Mobile Computing -Offloading through Data Size-Based Algorithm (DSBA)
Word Count:	5256
Page Count:	21

Signature:	Murugalakshmi
Date:	13th August 2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Optimizing Mobile Computing -Offloading through Data Size-Based Algorithm (DSBA)

Murugalakshmi
x21207232

Abstract

The computing power and battery sizes of handheld devices often pose difficulties in achieving optimal performance. The delegation of computationally intensive duties to more powerful remote servers or cloud infrastructures has proved to be a promising method for enhancing the performance of mobile devices.In this paper, we propose a novel Data Size-Based Algorithm (DSBA) to efficiently address the task offloading issue seen in mobile handsets.DSBA dynamically evaluates the computational complexity and data size of tasks to identify the most appropriate task for offloading, thereby minimizing the overhead and latency associated with remote execution. By taking into account the size of the data, DSBA seeks to optimize the communication cost and processing time, assuring a balanced trade-off between local execution and task offloading.The aim of the research is to offload mobile task with the minimum energy consumption and time ,At the same time to increase the number of bits processed. We shall see how this can be achieved locally or by offloading it to cloud platform.DSBA seamlessly adapts to varying network conditions and task characteristics making it robust to this real world of mobile cloud computing.

1 Introduction

Mobile Cloud Computing a paradigm that has transformed mobile devices with the power of cloud computing to perform many task on the go.This computation, storage, and processing task to remote, fog or cloud servers has empowered the efficiency to a great extent.However as the scope and complexity grow there is always a pressing need to enhance this technique. Noor (2018) refers to issues in offloading tasks in mobile devices. The experiments performed and the study suggests that optimizing high computational task and battery efficiency plays a key role in computing.

This paper presents a novel technique for offloadingtask on cloud computing resource provisioning that takes data size into account. Data size awareness is a multidimensional view of the data, including its storage, energy, and tasks, that provides information about the data's content, size, and activity, which we refer to as task queuing. As noted by Sani (n.d.), however, there have been numerous digital innovations in such outsourcing tasks, and there is still room for improvement. While using data-intensive applications on their mobile devices, users experience service delays due to network or storage issues. Keeping a large number of images/video files on a device causes an increase in power consumption and a decrease in battery life, which ultimately delays the device's performance.

We take into account high-end mobile devices with graphics cards, multicore processors, and an increase in the number of IoT device consumers.However, IoT devices have demonstrated the ability to manage such compute-intensive and data-intensive applications by offloading the task from devices to the cloud.The storage and processing of data is always a challenge for users; therefore, these tasks are offloaded to high-resource-rich cloud or fog nodes or performed remotely in order to improve performance and reduce battery consumption. The primary parameters considered during offloading are data size, task queue, and energy usage.

Research Question:

1. How can the performance of task offloading on mobile devices be effectively optimized with respect to factors such as dynamic resource allocation, network parameters, and energy consumption?

2. How data size awareness can efficiently offload mobile computational tasks on cloud for low energy consumption?

This novel technique of Data size based algorithm(DSBA) is designed for data-intensive task on mobile cloud computing applications. This suggested model will aim to efficiently offload task determining the data size while minimizing the completion time to achieve optimal performance and resources with a low energy

Structure of the Paper:

Following in the report is Section-2 Related work which will give an overview of the past research done in MCC whereas Section 3 will provide details of the methodology on this novel technique. Architecture and implementation is showcased under Section 4 , which is followed by evaluation and experiments under Section 5. Conclusion and Future work is also mentioned in further sections

2 Related Work

Mobile Cloud Computing (MCC) has gained significant attraction as a promising technology to improve the performance and efficiency of mobile applications.

2.1 Related work on Mobile Cloud Computing

Task offloading and resource allocation in Mobile Cloud Computing are critical challenges that directly impact the user experience and resource utilization. Taha Alfakih (2020) proposes an approach that employs deep reinforcement learning using the SARSA algorithm to optimize task offloading and resource allocation in Mobile Edge computing environments.
The author leverages the SARSA algorithm, which uses reinforcement learning method, to optimize task offloading and resource allocation in MEC. SARSA (State- Action-Reward-State-Action) is an on-policy temporal difference learning algorithm that collects from the interactions of an agent with an environment. By modelling the problem as a Markov Decision Process (MDP), the algorithm updates its policy based on the observed state

transitions, actions taken, and corresponding rewards received. In this context, Jia Yan (2020) SARSA is utilized to learn an effective offloading policy and resource allocation strategy that maximizes the overall utility of the MEC system. However, the study doesn't go into detail about the trade-offs between performance measures like delay, energy usage, and speed. Also, the study might not be able to evaluate how well the suggested method scales and works in large-scale mobile edge cloud applications.

Dynamic task arrivals, changing processing capabilities, restricted communication capacity, and judgements on work offloading are encountered by MEC systems and Liu and Mao (2016) has provided a detailed study on it. In the context of obtaining delay-optimality, a number of scheduling strategies, such as heuristic-based, game theory-based, and optimization-based algorithms, as well as machine learning methods, have been considered. Concerns around quality of service, edge-cloud cooperation, security, and privacy have also come into the spotlight. This study emphasis on the need of conducting empirical assessments and doing more research in order to solve newly developing difficulties and improve the effectiveness of job scheduling in MEC systems.
Rajkumar Buyya (2018) approaches are organized under the categories of task, data, and computation offloading, resource management, data compression, and enhancement of quality of service. The evaluation takes a comprehensive look at several methods that are considered to be state-of-the-art today and assesses both their advantages and disadvantages. It highlights difficulties in areas like as safety, energy efficiency, and the integration of new technology. The assessment highlights the need for continuing research to develop MCC, boosting system performance and user experience in mobile devices with limited resources.

Mohammad Yahya Akhlaqi (2023) explores the paradigm of task outsourcing in Mobile Edge Computing, focusing on current issues, adopted approaches, performance evaluation, and future directions. The article identifies obstacles in task offloading, including optimal offloading candidate selection, dynamic network conditions, privacy and security concerns, and peripheral server load balancing. The review examines numerous offloading techniques, including centralized and distributed offloading schemes, machine learning-based decision-making algorithms, and heuristics. In addition, Gilsoo Lee (2019) it assesses the efficacy of these methods, taking into account factors such as latency reduction, energy efficiency improvements, resource utilization, and overall system performance. In the future, the review suggests exploring novel offloading strategies, integrating AI/ML techniques, considering heterogeneous edge server architectures, and supporting emerging technologies such as 5G in order to further optimize the task offloading effectiveness in Mobile Edge Computing.

2.2 Related work on Offloading task on Mobile Device

MobiCOP, yet another outsourcing framework designed to address the limitations of existing platforms, is introduced in another research paper.MobiCOP, unlike most alternatives, optimizes entire long-running computation-intensive tasks rather than concentrating on minute refinements, providing up to 17x performance enhancements and up to 25x greater battery efficiency than local task executions. The client component of the framework is completely self-contained in a library and compatible with the vast majority of stock Android devices, facilitating integration, while the server component is readily available via

a public AWS AMI for rapid deployment. The Rajkumar Buyya (2018)Guillermo Valenzuela (2018) architecture is concerned with transmission speed and network latency but disregards the data capacity of the task to be completed. Here, it leaves some questions for the 2018 research to answer.
There is another study that explores the framework design, implementation, and evaluation in various scenarios, emphasizing its impact on user experience, energy efficiency, and system performance.Amir Vahid Dastjerdi (n.d.) discusses the advantages of context–awareness to enhance the task offloading's efficiency and to determine the potential challenge. However, there is a scope for enhancement for different cloud instances on this proposed approach

MTFCT (Multi-Tier Fog-Cloud Task Offloading) approach, a task offloading technique designed by Rajni Jindal (2020) to optimize resource utilization and reduce latency in Fog Computing and Cloud Computing environments. This article examines the design and implementation of MTFCT, its application in both fog and cloud scenarios, as well as its impact on system efficiency, resource utilization, and latency reduction. The MTFCT strategy utilizes both fog and cloud resources for efficient data processing, providing benefits such as low-latency and localized processing in Fog Computing and scalability in Cloud Computing. The review assesses the performance of MTFCT in various real-world scenarios and identifies potential challenges and research directions for augmenting its efficacy in dynamic Fog and Cloud Computing environments. However this complete study was theoretical and no real-time experiment was performed in any cloud instance or fog node

Ibrahim Alghamdi (2019) examines time-optimized task outsourcing decision-making in Mobile Edge Computing, concentrating on existing approaches, performance evaluation, challenges, and future directions. It determines the decision-making algorithms that seek to minimize task execution time and latency in outsourcing decisions, taking centralized and distributed approaches, machine learning-based algorithms, and heuristic methods into consideration. In this work, Meixia Tao (2019) shows how the Gibbs sampling method produces a good offloading decision. This study is based on a Mobile edge computing system with multiple clients that employs a technique that requires a task input on a single wireless device for the final output from numerous additional wireless devices for a ideal threshold offloading. The Gibbs sampling technique produces a decision outputSuzhi B (2019) proposal is an ADMM, or a massively rotating orientation technique, which addresses the complexity issue in huge networks. Decomposition is an ADMM under which the initial optimization deteriorates problems into several concurrent comment threads. Additionally, the report discusses the Integration of RF-based power transmission and edge computing without a connected setup.It can likely address the IoT networks' performance issues and restriction

The research paper from 2021,investigates existing research and methods for designing effective task-offloading algorithms for cloud computing. This review examines dynamic multi-objective evolution techniques, taking into consideration performance, resource utilization, energy efficiency, and load balancing, in order to pave the way for a novel and effective algorithm that optimizes the allocation of tasks in cloud environments. Su Hu (2021) demonstrated time-based performance-based experiments which are proved theoretically in the study.Similar study in performed by Junwen Lu (2022) where the entire research is in the field of energy-energy-efficient task offloading and scheduling in mobile cloud computing. The architecture majorly includes the 4 components such as the mobile scheduling manager, the cloud scheduling manager, cloud resources, and jobs. **?** decision engine checks the task based on the mobile network and bandwidth to offload them remotely or in the cloud server. Even after successful research, the study has space to work on data size and energy consumption of the task.

3 Methodology

Many researchers have presented their work on mobile task offloading due to which there is a wide range of potential scenarios. However, this section provides an in-depth analysis of the methodology used to assure the success of this novel research. The following also elaborates on the tools and technologies utilized to accomplish the desired results. We are making use of a custom algorithm ie the DSBA to evaluate the task and make a crucial execution decision. This decision-making engine focuses on determining whether the task will be executed on the local device, i.e. the mobile phone, or in the cloud. Researchers have already demonstrated the same on Fog nodes mentioned above in the related work. The results acquired from either method are displayed on the mobile device's display, allowing for additional analysis, research, and conclusive conclusions. The methodology is designed to maximize efficiency and effectiveness throughout the research process by optimizing task execution and resource allocation. Various analytical tools and cutting-edge technologies are used to support the algorithm's functionality, allowing for seamless integration and robust decision-making capabilities.

3.1 System Model

A mobile client that utilizes MCC architecture consumes lot of cloud technologies in order to run its applications in a smooth manner on cloud infrastructures via mobile Web-based services.The importance of this model is to largely focus on the data size of the task and decide on the offloading approach.The two primary goals of these designs are to maximize the use of available resources on the device while simultaneously maximizing performance efficiency. Some concerns are currently being explored at various levels of MCC designs in order to enhance the execution efficiency of mobile applications while keeping costs to a minimum.

3.1.1 Major components of this proposed system

This proposed algorithm largely consists of these three components

1. **Data Manager** :The Data Manager is responsible for managing proficiently the duties generated by mobile devices. It manages the allocation of computational resources for task execution and processes the collection of data-related task. The Data Manager is aware that executing tasks on mobile devices can be resource-intensive, necessitating significant energy consumption, computational power, and time.To effectively manage these responsibilities, the Data Manager employs strategies that optimize resource utilization Data Manager employs techniques such as task

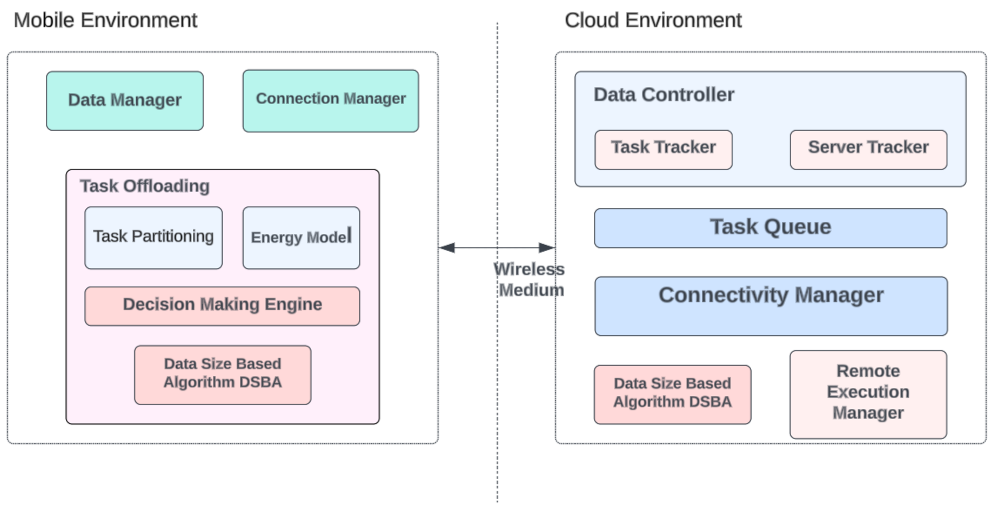


Figure 1: Architecture of the DSBA Model

aggregation, load balancing, and task outsourcing to accomplish efficient task execution.

2. **Data Controller** : This model takes into factors like the purpose of the job, the state of the network conditions, and the resource availability. Tasks are chosen based on how well they can be offloaded and then split up into components that can be transmitted. To make connections as efficient as possible, data will be preprocessed and compressed. Offloading choices will be based on parameters like task execution time, energy consumption, and available bandwidth, aiming to minimize overall execution cost. The design model for data sharing demands a strong way to make decisions as well as effective ways to communicate to each other so that mobile devices and cloud resources can work together smoothly.

3. **Task Queue** : The task queue plays a crucial role as a connecting component in a cloud-based system that is responsible for keeping track of incoming tasks from mobile devices, ensuring that they're completed. The system effectively manages and pulls tasks from a designated queue, distributing them to server resources that are currently accessible, taking into account several aspects like task priority and resource availability. The queue operates within a specified capacity and employs various strategies, such as dismissing incoming tasks when it reaches its maximum capacity or shifting jobs to other servers, in order to mitigate congestion. Load balancing systems are used to provide equitable allocation of tasks, hence enhancing the utilization of resources and optimizing response times. The task queue plays a vital role in enabling the regulated and effective execution of tasks, as it strategically makes critical choices to ensure stability and optimize performance within the cloud environment.

3.2 Computational Algorithm

The presented Data Size-Aware Offloading Decision Algorithm facilitates the dynamic offloading decision process for resource-constrained applications.It will accept task such as HD video streaming for an example , resource provider as inputs and produces a distinction between tasks to be executed locally and those eligible for execution in the cloud. The algorithm begins by gathering data on the availability of local and cloud resources . It will then determines the data size associated with each task. The algorithm evaluates network connectivity and resource availability for jobs with data sizes greater than or equal to 120 MB (this is for an example). If both conditions are met, it employs partitioned application-specific methods and identifies resource dependencies. It considers the data size algorithm to a subsequent matrix for task partitioning, resulting in a set of tasks designated for local execution and another set designated for outsourcing to the cloud. If network or resource conditions are unfavorable, on the other hand, tasks are executed locally. Overall, the algorithm focuses to optimise task allocation in heterogeneous computing environments based on data size, network state, and resource availability, thereby improving the efficacy of deploying decisions.

Algorithm 1 Data Size Based Awareness Algorithm

```
Require: Application, Resource Provider
Ensure: Tasks executed locally and tasks to be executed in the cloud
 1: resource_provider ← Get local availability, cloud availability
 2: application ← Get the application
 3: data_size ← calculate data size of the task
 4: if data_size $\geq$ 10 MB then
 5:    check network state
 6:    if network available then
 7:       check resource provider
 8:       if resource provider available then
 9:          find the partitioned methods for the application
10:          find the dependencies between resources
11:          find the Offloading platform
12:          task Local-Result ← Local
13:          task Offload ← Offload
14:       else
15:          execute locally
16:       end if
17:    else
18:       execute locally
19:    end if
20: else
21:    execute locally
22: end if=0
```

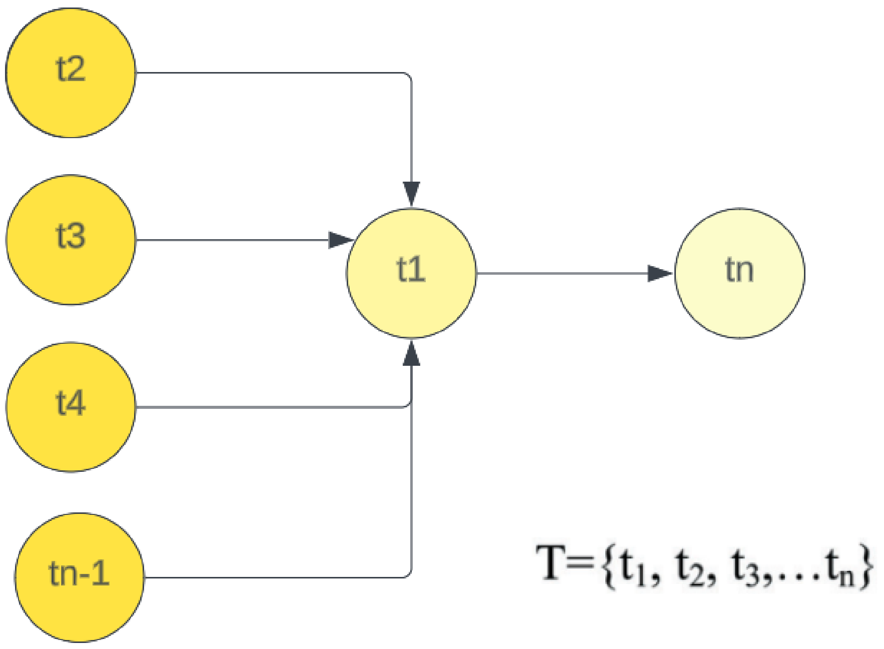


Figure 2: Task division

4 Design Specification

For the research $T = t1, t2, t3, ldots, tn$, where $T = t1, t2, t3, ldots, tn$, is considered to be partitioned into n offloadable tasks for the purposes of this research. Each task t_i will have a distinct load and require separate computational resources. Which task must execute in the mobile environment and which task must be offloaded to the remote environment for computation is the primary concern in task offloading. Figure 2 demonstrates that the project T is subdivided into n tasks. Consider that the task t_1 requires output data from tasks t_2 to $t_n - 1$ in order to execute, and that t_1's output is provided as input to task t_n.

To address the issue mentioned earlier, the Data size model has been created, which focuses on the amount of energy required for each task. Based on the task's properties, the decision-making engine determines whether the task will be offloaded or executed locally.

$delta(t)$ represents the burden of each task in this implementation. Consider $varphi(t) = 0, 1$ to be a decision variable; if $varphi(t)$ is set to 0, it indicates that the task can be offloaded, whereas if it is set to 1, the task is executed in the local environment.

4.1 Local Environment - Energy Consumed

The local energy consumption equation is as below:

$$\varepsilon_{\text{local}}(t) = \sum_i P_{\text{local}} \times \tau_{\text{local}}(t) \tag{1}$$

The local completion time equation is as below :

$$\tau_{\text{local}}(t) = \sum_{i} \delta_i(t) \times \varphi_i(t) \times C_{\text{local}}^{-1} \tag{2}$$

If the task as per the above 2 equation goes to zero then the battery energy consumption will be go to zero in the local environment. The results for this can be seen in the FireStore database in the Evaluation section with experiments performed.

4.2 Cloud environment - Energy consumed

The equation provided represents the time required for task completion when offloading tasks to the cloud environment.It takes into account various factors, including the task index i, the input data size I_{d}, the transmission bandwidth T_{b}, the workload $\delta_{i+1}(t)$, and the decision variable $\varphi_{i+1}(t)$ determines if the task needs to be executed locally or offloaded. Moreover, the computation speed C_{cloud} of the cloud environment is considered under this calculation. This comprehensive formula provides insights into the time of task offloaded in cloud computing .

The time taken to complete offloading in cloud environment is given by:

$$\tau_{\text{cloud}}(t) = \frac{i+1}{I_{\text{d}} \times T_{\text{b}}} + \delta_{i+1}(t) \times (1 - \varphi_{i+1}(t)) \times C_{\text{cloud}}^{-1} \tag{3}$$

4.3 Overall Energy consumption

The overall energy consumption for the above can be given as below

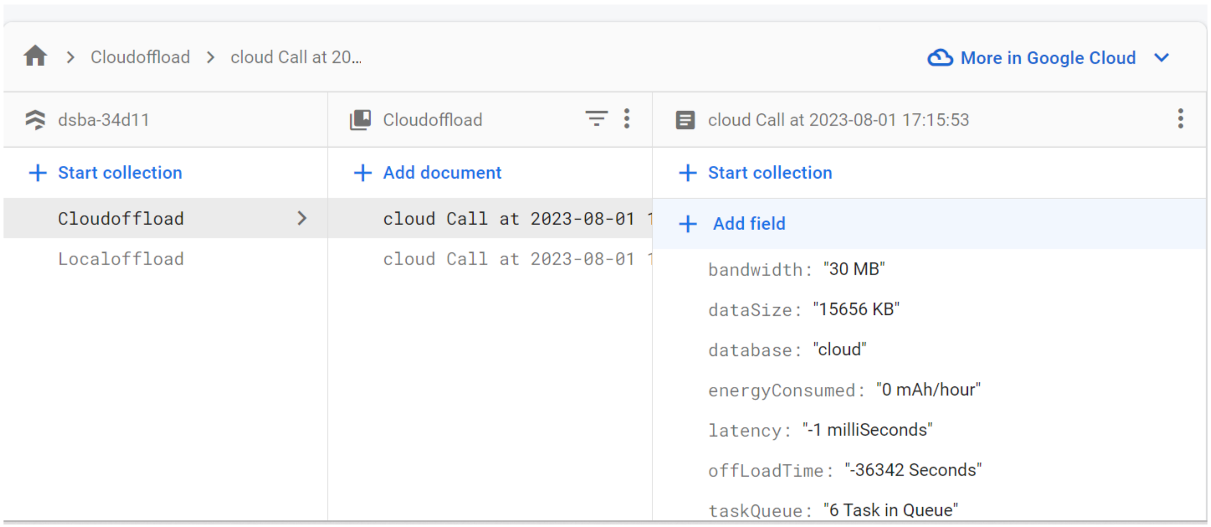
$$\varepsilon_{\text{cloud}}(t) = (i+1) \times P_{\text{idle}} \times \tau_{\text{cloud}}(t) \tag{4}$$

4.4 FireBase Parameters

The figure 3 represents an overview of the parameters and equation calculation based on the above algorithm. The Firebase plugin, a potent toolset that enables an extensive range of additional functions, is considered as it can easily linked with our Android application. Google's Firebase service is tightly linked with its Google Cloud server. In the context of this study, Firebase is a crucial tool for meeting the needs of data storage and performance monitoring. As a crucial component of the Google Cloud service portfolio, Firebase not only guarantees safe and scalable data storage but also improves the entire functioning of the application

5 Implementation

The decision-making engine is one of the main offloading modules in the proposed design. The purpose of the decision-making engine is to determine whether a task should be offloaded to the cloud environment or completed locally. It describes task outsourcing that



```
bandwidth: "30 MB"
dataSize: "15656 KB"
database: "cloud"
energyConsumed: "0 mAh/hour"
latency: "-1 milliSeconds"
offLoadTime: "-36342 Seconds"
taskQueue: "6 Task in Queue"
```

Figure 3: Cloud and Local Parameters

Table 1: Nomenclature and Descriptions

Symbol	Description
$\delta(t)$	Workload of the task t
$\varphi(t)$	Decision variable
τ_{local}	Completion time of task at mobile environment
τ_{cloud}	Completion time of task at cloud environment
C_{local}	Computation speed of mobile environment
C_{cloud}	Computation speed of cloud environment
$\varepsilon_{\text{local}}$	Energy consumption of task in the mobile environment
$\varepsilon_{\text{cloud}}$	Energy consumption of task in the cloud environment
$\varepsilon_{\text{network}}$	Energy consumed by the wireless medium
P_{local}	Power consumption of the mobile environment
P_{idle}	Power consumption at the idle state of the mobile environment
I_{d}	Input data to be offloaded
O_{d}	Output data to be received by the mobile environment
T_{b}	Transmission bandwidth of the network

prioritises completion time and energy consumption above all else. Job execution in a mobile environment must satisfy the user's needs. Consequently, it must meet the user-specified deadline hence whenever the user hits the DME the offloading code is triggered to make the decision based on the data size of the video streaming.

5.1 Tools and Technologies Used

The implementation heavily relies on the incorporation of multiple asynchronous functions, which are instrumental in ensuring the seamless execution of the application's tasks while concurrently managing secondary tasks. This combination of specific development tools and environment specifics results in an application that is designed to operate effectively in terms of both its technical infrastructure and its user-facing features.

- **Android Studio**

  The fundamental Android application serves as the basis for the experiment, with its implementation on the Android emulator it allows seamless connection to the application's underlying source code. Even though the application's APK can be easily installed on a tangible mobile device for testing purposes, the emulator is used here for this experiment. Android Studio Code, specifically version 2022.2.1 Patch 2 Flamingo, has been used as the main environment for development. Integral to this configuration is the incorporation of a 64-bit server Virtual Machine supported by the OpenJDK Virtual Machine. We have used the Pixel3a emulator for application deployment, which simulates the characteristics of the intended device. This virtual device is set up to utilize Android API 34 Google APIs. Importantly, the application's architecture is supported by version 8 of the JAVA programming language. The application is launched on Google Pixel API extension level 7 x86 64.

- **Firebase Storage**
  The fact that Firebase serves as a cloud-based storage medium for crucial data produced during the algorithm's execution is important. Two tables, the "Offload Records" table and the "Firebase Database" table, are systematically kept in order to support thorough record-keeping. The former is meant to carefully record each choice the algorithm produces. The table has parameters like bandwidth , latency, time consumed, battery level and so on.This table serves as a visual depiction of execution time on both local and cloud platforms, providing insight into the effectiveness of the algorithms. Figure 3 above depicts the same.

- **GitHub**
  The code is pushed to GitHub to control the version and easy deployment of CI /CD pipeline to make most use of the AWS Web services for cloud platorm.

- **AWS Cloud Service** For the AWS Services we use Lambda function to integrate with the cloud instance , API gateway post resource action is helps trigger the lambda function.Figure 4 gives an sight of the Lambda function used to trigger the API Gateway

On the other hand, the "Database" table functions with greater precision. Its architecture centres on retaining a single record for each task, with changes being initiated by

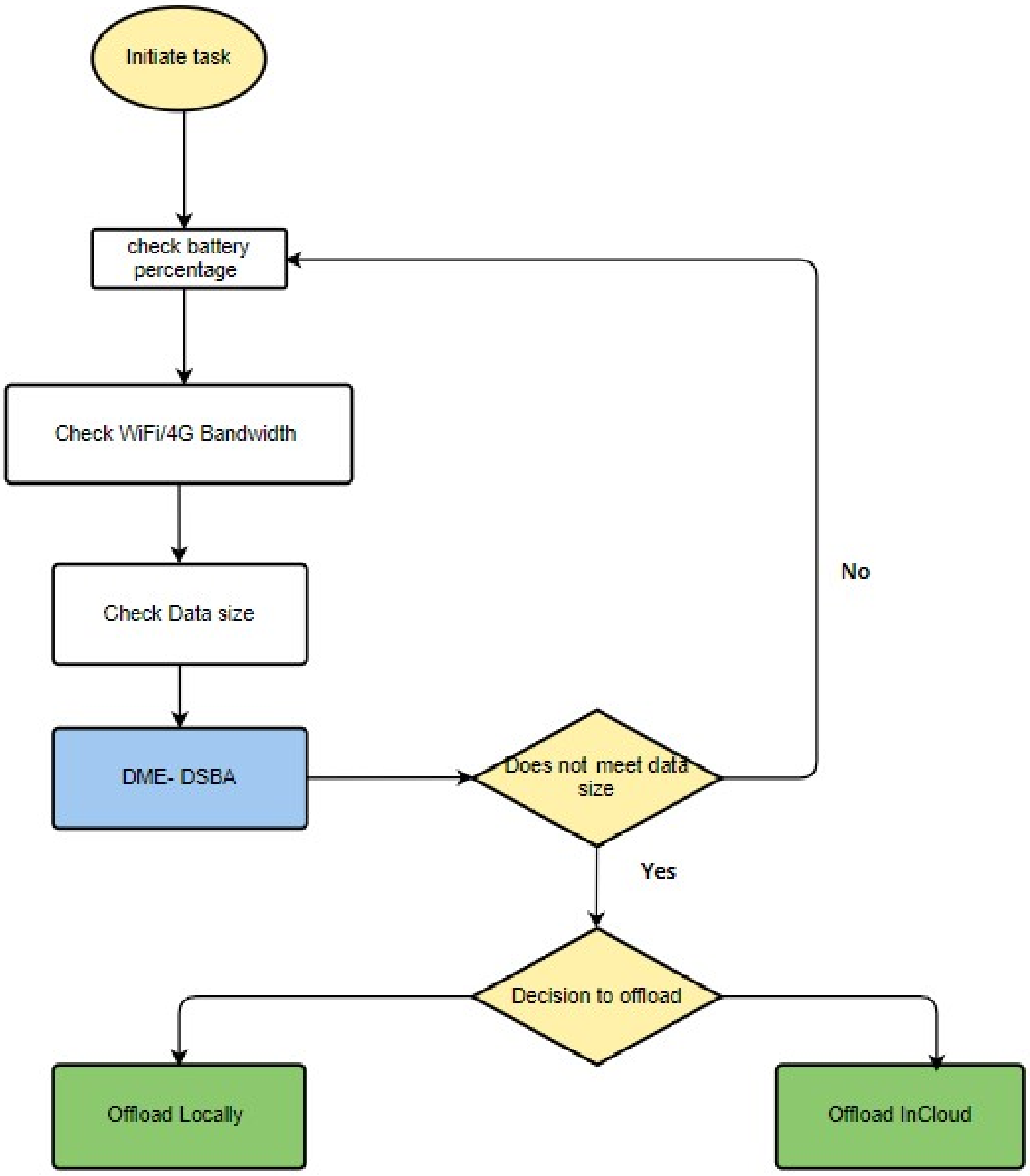
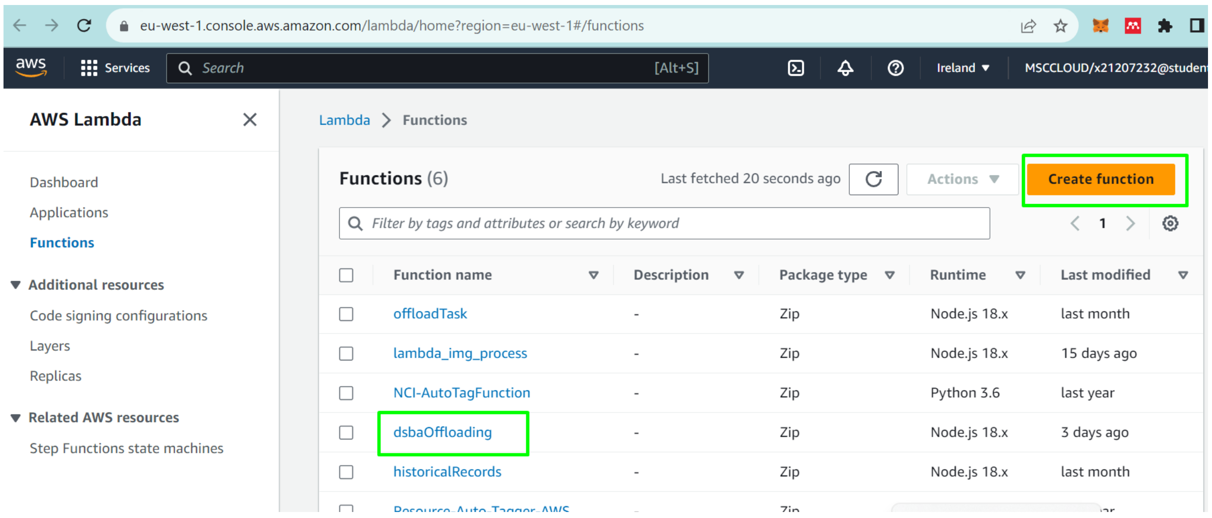


Figure 4: Decision Engine - Offloading process flow


eu-west-1.console.aws.amazon.com/lambda/home?region=eu-west-1#/functions

aws Services Search [Alt+S] Ireland MSCCLOUD/x21207232@studen

AWS Lambda

- Dashboard
- Applications
- Functions

Additional resources

- Code signing configurations
- Layers
- Replicas

Related AWS resources

- Step Functions state machines

Lambda > Functions

Functions (6) Last fetched 20 seconds ago Actions Create function

Filter by tags and attributes or search by keyword

Function name	Description	Package type	Runtime	Last modified
offloadTask	-	Zip	Node.js 18.x	last month
lambda_img_process	-	Zip	Node.js 18.x	15 days ago
NCI-AutoTagFunction	-	Zip	Python 3.6	last year
dsbaOffloading	-	Zip	Node.js 18.x	3 days ago
historicalRecords	-	Zip	Node.js 18.x	last month
Resource-Auto-Tagger-AWS	-	Zip		

Figure 5: AWS Lambda function

either a change in performance or by the algorithm's identification on the data size for subsequent executions to offload task. This table's basis is made up of vital information including bandwidth, latency, time , performance metrics, and details about the execution environment.Figure 3 can be referred for the same. Significantly, Firebase's powerful capabilities go beyond storage and provide information on how much of the CPU is being used overall when an application is being run. Additionally, it offers graphical visualizations of changes in reaction times for the program, adding to a thorough analysis of its performance dynamics.

Virtual Machine (VPC)	
CPU	Quad-core processor
Memory	16GB
Cores	12

Table 1: Virtual machine Configuration

Deeply integrated into Google's cloud architecture, Firebase emerges as a crucial partner, enhancing the application's effectiveness with advanced data management and analytics capabilities. Its seamless integration strengthens the data-centric aspects of the study, assuring the best use of resources and producing insightful data that aids in making well-informed decisions.

5.2 Algorithm 1 - Task execution

The above pseudo-code is all about the decision-making engine on how the task execution will happen either on the cloud or local environment based on the data size of the task. The set of tasks with the execution time (t) is the input.

```
Input: $T = \sum_{i=1}^{n} t_i$

Output: task execution in cloud/mobile environment

Begin

Step 1: if $\delta(t) \leq 0$ and $\varphi(t) \leq 0$ and $\varphi(t) \geq 1$ then

            Go to step 4

Step 2: for all $\delta(t)$, if($\varphi(t)$==1)

                    Calculate $\tau_{local}$;

                    If $\tau_{local} \leq \tau_{min}$ then

                        Calculate $\varepsilon_{local}$;

                    Otherwise

                        Go to step 3

                    End if
```

The algorithm will first check if the condition related to task characteristics (δ and ϕ) is met, If yes then avoid offloading the task which is mentioned under Step 1. If No then it will decide to execute locally based on Step 2. If not suitable then the algorithm will calculate the performance of executing the task in the cloud and decide if offloading is needed or not under Step 3. If none of the above conditions are met then the task will not be offloaded as per Step 4 letting the optimal execution environment for each task based on the factors of resource availability and efficiency.

```
If $\varepsilon_{local} \leq \varepsilon_{min}$ then
        Execute $t_i$ in local mobile environment;
                Otherwise
                        Go to step 3
                End if
        End for
Step 3: if ($\varphi(t)$==0)
            Calculate $\tau_{cloud}$ and $\varepsilon_{cloud}$;
            Offload $t_i$ to the cloud server;
            Break;
Step 4: Return “do not offload”;
            Break;
End
```

5.3 Algorithm 2 - Energy Consumption

Figure 6 of of Pseudo code computes the overall energy consumption based on the decision engine. If the decision is to "offload," the energy consumption of the task in the cloud is calculated. If the decision is not to "offload," the energy consumption of the task in a mobile environment is calculated and the task is executed. Finally, it calculates the total energy consumption (T) of the activity by combining the energy consumption from both environments.

The below **flowchart** under Figure: 4 represents the working of the decision engine. On HD video streaming say of data size 4MB consumes is played, the bandwidth is of 2-5 Mbps and the playback time be 60-120 seconds , battery percentage as 70 percent ,under 4G network. The Decision engine will determine the data size required to reduce the energy and the latency deciding to offload it remotely on the handheld device or send it to Cloud platform with the resource utilization. The Data size is taken into account by the engine and it the size condition matches then the decision to offload is made on either cloud and remotely. Offloading under cloud platform reduces the latency and energy consumption for better resource utilization of the android device.

```
Input: Decision from Algorithm 1
Output: Overall energy consumption of the Task
Begin
• Step 1: If Decision = offload then
    – Calculate energy consumption of $t$ in cloud
    – Execute $t$
• Step 2: If Decision $\neq$ offload then
    – Calculate energy consumption of $t$ in mobile environment
    – Execute $t$
• Step 3: Combine the energy consumption of both environments
    – Calculate $\varepsilon(T)$
End
```

Figure 6: Algorithm 2

6 Evaluation

Under this session, the proposed architecture is evaluated and analyzed in order to check the performance of the device with respect to the parameters such as energy consumed and completion time taken to perform a high computational task. For this experiment, we have emulated the results using android studio's emulator which is Google Pixel and Pixel Pro. Different emulators are used to ensure the results perform on various types.

6.1 Energy Consumption

Over the testing performed on the virtual machine , it is observed that the decision engine determines the size to offload the task either locally or on cloud. Meanwhile it is seen the energy for data size ranging from 7MB of video data to 15GB.Considering an example of Netflix which streams HD video , the program uses around 1GB data for every 60minutes video.With respect to this the battery consumed goes upto 24 percent on the streaming for about 30 minutes on 4G data for a HD video of 1080p . On the otherside the same streaming when offloaded to cloud is observed to consumed more on 4G and a difference of about 10 joules less on Wifi.

6.2 Time Consumption

On testing with the proposed algorithm , the power consumption of the CPU on the local mobile environment and the cloud is shown in the below graph.It is shows that the by offloading the task based on the data size will result in lower energy consumption on the

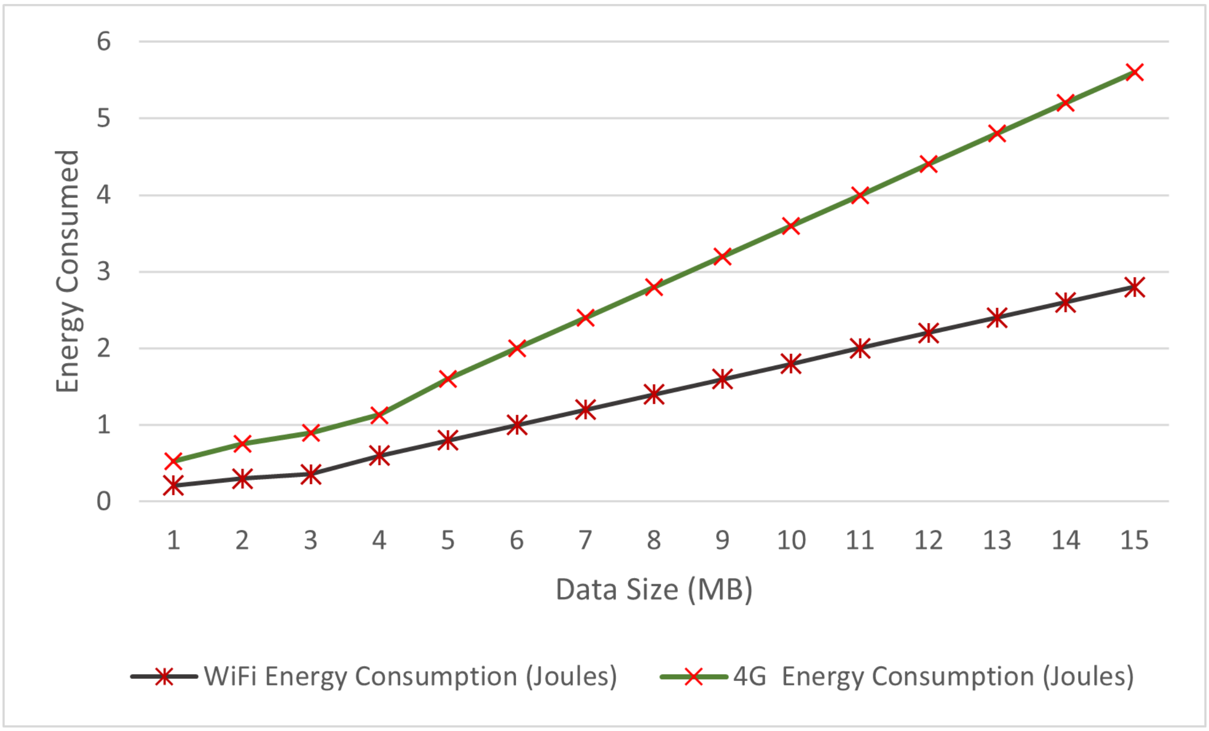


Figure 7: Energy consumption

local device. This ultimately shows reduced CPU and memory utilization. The hardware and network parameters are subject to consideration for efficiency.

6.3 Overall Performance Observation

Graph 8 represents the overall battery and time consumed on different networks to offload the HD video streamed by the user.IT is observed that the offloading over cloud platform lets the mobile performance better with its low energy consumed. This showcase the better resource utilization of the device which ultimately directs to better user experience and more task performance. Figure 9 depicts the CPU performance and the memory of the mobile device on emulating the offloading task.The testing is performed both on Wifi and 4G network with 1080p Video Streaming from 60 seconds to 1800 seconds.

6.4 Discussion

In our research project, experiments using the DSBA method are largely focused on examining the possible advantages of taking data size into account while offloading workloads for demanding computing activities to mobile devices. Researchers lack much attention to this specific issue until now. While earlier research, like that of Sani (n.d.), has mostly concentrated on elements like bandwidth and latency, the topic of data size is still largely unknown. We tried to close this gap by proving that data size may be a crucial variable in the world of mobile computing via the course of our project.

The testing results show significant reductions in energy usage and execution times, particularly in circumstances which is required by high demanding activities like HD

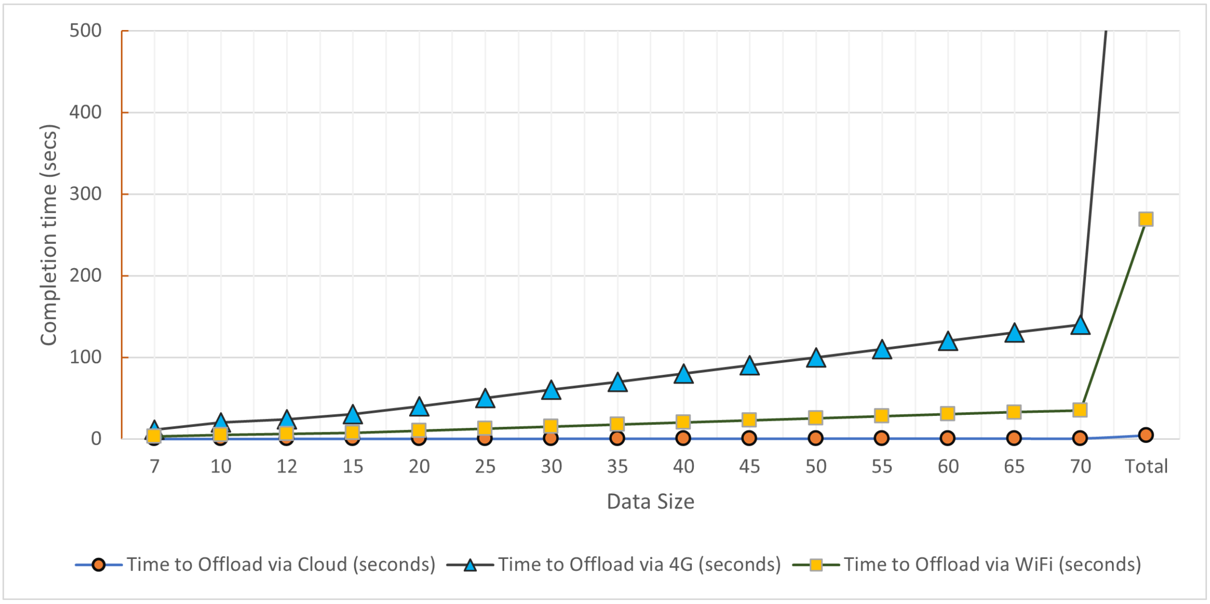


Figure 8: Offload time

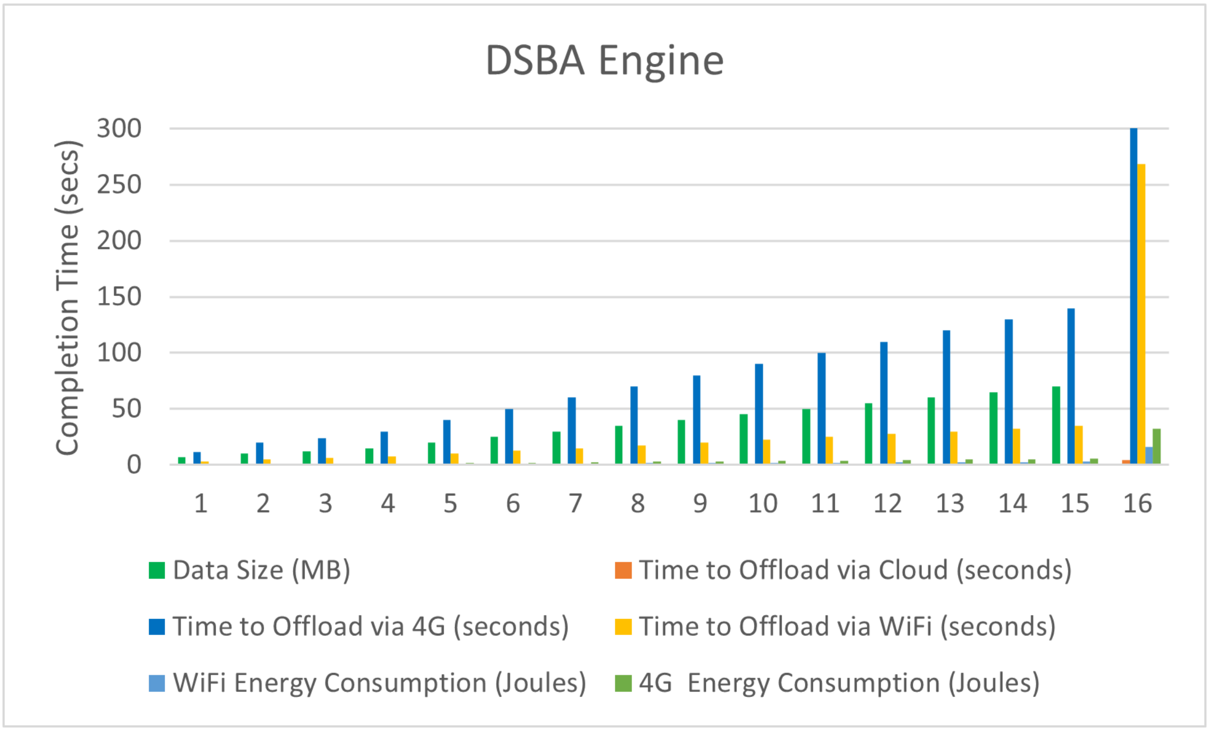


Figure 9: Overall Energy Consumption

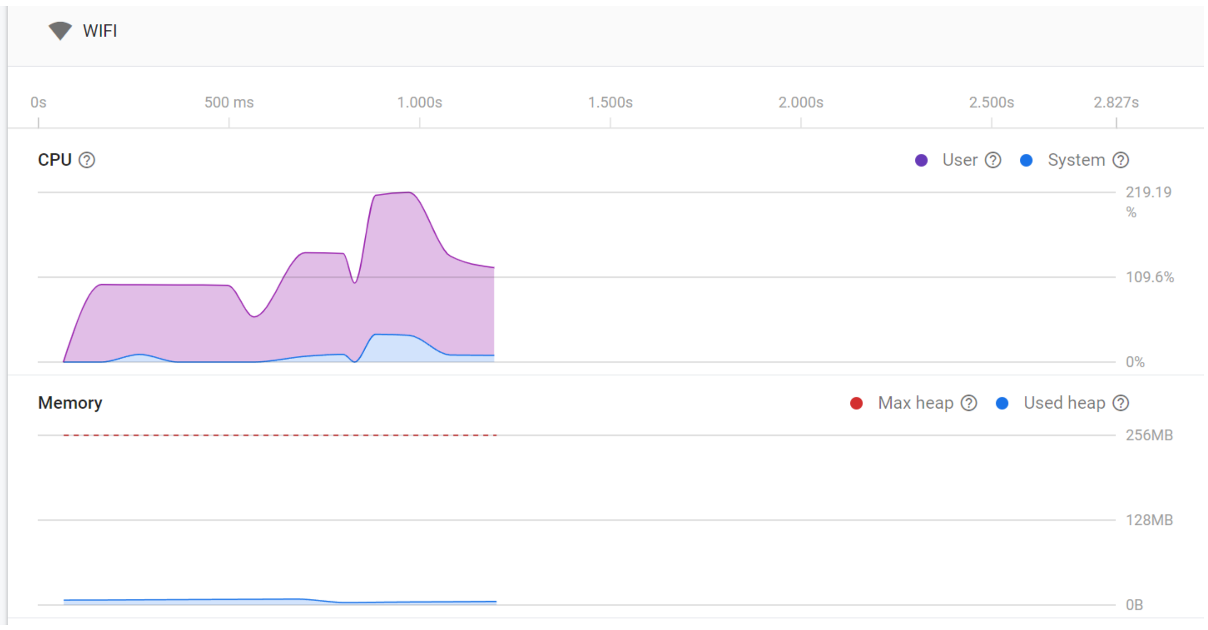


Figure 10: CPU- Performance

video streaming or high end gaming. Although we are showcasing the emulator from the virtual studio in this report, the experiments can be used in Android smartphones for testing. The efficacy of the experiment has been improved taking into account the data-size which was not considered in the research Sani (n.d.), However we have largely taken into account the network parameters referred to in the above research.

We also explored trials on fog computing, which suggest offloading to edge devices as a viable remedy. These fog-based offloading methods do, however, have certain limitations, Their effectiveness can be called into doubt by elements like network connection and Heterogeneous environment. Fog devices are also inherently vulnerable to security flaws because of their edge-of-the-network location, especially when it comes to sensitive data. Our innovative method, which is based on approaches for offloading computing chores depending on data size, results in a quicker and more effective solution. The program shows the flexibility to react dynamically to changing network circumstances across numerous connection choices, such as 4G, WiFi, and LTE. This flexibility guarantees the best decision-making depending on the capabilities of the gadget in certain situations.

Our experiments demonstrate that the smaller data tasks can be executed locally while offloading bigger data tasks to the cloud. This careful balancing decision can effectively be done by better resource utilization, latency reduction, and low energy consumption. Users can thus have faster reaction times and shorter wait times while doing data-intensive operations. This experiment showcases how a data size-aware method can significantly improve process efficiency and support the importance of data processing within the context of offloading task.

7 Conclusion and Future Work

In conclusion, our research effort uses the novel DSBA method to investigate how data size affects task offloading for computationally intensive tasks on mobile devices. While this feature has often been disregarded in other research, this testing clearly demonstrated its critical significance. Our application proves its usefulness in boosting overall efficiency and time-consciousness by taking into account data size along with elements like bandwidth and latency. We have shown via extensive studies that data size awareness may significantly reduce execution time and energy usage, especially for energy-intensive workloads like HD video streaming and high-end gaming. Our algorithm's capacity to constantly adjust to changing network circumstances and device capabilities further reinforces its usefulness. As a future work we can consider developing adaptive load balancing mechanisms that can distribute the offloading tasks based on both data size and computational resources on different network conditions.This can enhance the user-experience to adapt in the dynamic environments.

References

Amir Vahid Dastjerdi, R. N. C. (n.d.). A context-aware offloading framework for heterogeneous mobile cloud.

Andrew (2018). Mobile task offloading based on bandwidth and battery availability.

Gilsoo Lee, W. S. (2019). An online optimization framework for distributed fog network formation with minimal latency.

Guillermo Valenzuela, P. S. (2018). Mobicop: A scalable and reliable mobile code offloading solution.

Ibrahim Alghamdi, C. A. (2019). Time-optimized task offloading decision making in mobile edge computing.

Jia Yan, Y. J. Z. (2020). An online optimization framework for distributed fog network formation with minimal latency.

Junwen Lu, Yongsheng Hao, K. W. Y. C. Q. W. (2022). Dynamic offloading for energy-aware scheduling in a mobile cloud.

Liu, J. and Mao, Y. (2016). Delay-optimal computation task scheduling for mobile-edge computing systems.

Meixia Tao, Jia Yan, Y. J. Z. (2019). Optimal task offloading and resource allocation in mobile-edge computing with inter-user task dependency.

Mohammad Yahya Akhlaqi, Z. B. M. H. (2023). Task offloading paradigm in mobile edge computing-current issues,adopted approaches, and future directions.

Noor, T. H. (2018). Mobile cloud computing: Challenges and future research directions.

Rajkumar Buyya, B. Z. (2018). Augmentation techniques for mobile cloud computing: A taxonomy, survey, and future directions.

Rajni Jindal, Neetesh Kumar, H. N. (2020). Mtfct: A task offloading approach for fog computing and cloud computing.

Sani, M. A. B. (n.d.). Dme: Technique for computation offloading in mobile cloud computing.

Su Hu, Y. X. (2021). Design of cloud computing task offloading algorithm based on dynamic multi-objective evolution.

Suzhi B, Y. J. Z. (2019). Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading.

Taha Alfakih, M. M. H. (2020). Task offloading and resource allocation for mobile edge computing by deep reinforcement learning based on sarsa, *Concurrency and Computation: Practice and Experience* **8**: 11.