National College of Ireland

Delay vs power consumption in edge/fog computing

MSc Research Project
MSc in Cloud Computing

Rakesh Singh Rawat
Student ID: 18190791

School of Computing
National College of Ireland

Supervisor: Manuel Tova-Izquierdo

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Rakesh Singh Rawat
Student ID:	18190791
Programme:	MSc in Cloud Computing
Year:	2020
Module:	MSc Research Project
Supervisor:	Manuel Tova-Izquierdo
Submission Due Date:	17/08/2020
Project Title:	Delay vs power consumption in edge/fog computing
Word Count:	7293
Page Count:	21

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

I agree to an electronic copy of my thesis being made publicly available on TRAP the National College of Ireland's Institutional Repository for consultation.

Signature:	
Date:	27th September 2020

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.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

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

Delay vs power consumption in edge/fog computing

Rakesh Singh Rawat
18190791

Abstract

With the advent in cloud technologies a parallel rise has been seen on the use of IOT infrastructure. As part of this Edge/Fog computing paradigm we see devices like smartwatches, mobile phones and other devices constituting it. For issues like computational power, limited storage, limited power/battery and non scalable architecture are already known to be a part as deficiency in resource department of in Edge computing. One viable solution to tackle this resource problem is to offload the workload from edge/mobile to either a fog/cloudlet server or to a centralised main cloud server. So running workloads locally will drain the limited resources, and entirely executing workloads on the cloud which is called cyberforaging and has a drawback that it will incur communication cost and will have a delay as well. So a balanced approach among these two is needed. I am proposing a Novel approach for this yet open challenge of choosing what to offload to cloud and in what order for different conditions faced by a resource constraint edge/mobile device having different priority workloads. So, choices will be either to face delay by offloading to a cloud and incur communication costs compromising QoS of application or to execute this workloads on the edge/mobile devices which is power/energy(resource) constrained device. Will use a priority based queueing scheme for marking workloads with priority. Using EdgeCloudSim as a simulator to reproduce environment to prove the novel approach. After experimentation I got results which shows a lower failure percentages of high priority applications, thus increasing there QoS and also dealt with the trade off dilemma.

1 Introduction

1.1 Overview

Need of computing, storage and resources is increasing everywhere with the advent of technologies like big data and IoT which is internet of things in parallel to and also in tune with the current cloud paradigm. Day by day we can see the applications in use within edge paradigm are getting more and more complex and so require more power in computing terms and storage too, also in terms of velocity, volume, and variety of data it is huge in these IoT devices. A drawback of cloud infrastructure which provides scalable, shareable resources at a very low cost is incurring transfer costs and latency while fetching or offloading data/code to cloud. And later the signals/results which is processed is being sent back to the IoT devices and that's where the latency comes into play is even more critical with some applications like medical/health field applications, real time apps or apps used in vehicles. Now for offloading we use Edge layer to a different device/sensor which works as a alternate computing server as a point near to where data is being

collected and will result in less latency. Also we can reduce latency by offloading code to fog computing layer or cloudlet (a small data center). For this particular case where we have a edge device having limited resource for example having a battery in a device which drains over time. Now this power constrained device because of heavy computation it may offload to save the power to a upper layer as fog/cloudlet layer or a centralised cloud layer but at the same time it may face a delay for receiving results from this offloading. So the concept of cyber foraging also comes into play in which the the device always offloads and abuse the feature of offloading and is not at all a utilised solution of using cloud layer with the edge layer having IoT devices or sensors (or even mobile devices). We need to achieve this balance of use of computing and storage resources of the cloud and edge. So we can find these power constrained devices mainly in Edge and sometimes Fog layer as well. The power consumption we are talking about is varying in nature as the device face network changes as it keeps on moving and consumption of wireless technology like WiFi or 4G or 5G are the parameters as communication overhead. I am researching in the area of this power/battery constrained devices which face the decision making problem of offloading the processes/workloads to another edge device part of edge computing or to a main cloud computing layer taking in consideration the delay faced for this offloading decision versus the drain of power if the workload is not offloaded and processed on the originating device itself, also the order of this offloading too needs to be prioritise in case of multiple workloads. Many papers discuss resource management techniques over managing this transition of data/workload to cloud or to another edge server and also to minimise the overall cost to user while doing so. Gill et al. (2019) Shekhar et al. (2020) Abbasi et al. (2020) Wu et al. (2019) Cui et al. (2019) Weijian et al. (2018) Karamoozian et al. (2019) Lin et al. (2019) Parameters considered by these numerous papers in algorithms purposed by them considers mainly the cost of transportation and energy constraint in another edge device and also the overall monetary effect of this faced by the user or organisation.
"Delay in offloading to cloud versus power consumption in resource constrained device?" Lan et al. (2019) My research deals with vertical distribution of processes from edge and Fog layer to cloud servers using the priority of the computational workloads using a Novel proposed approach.

1.2 Value and Contribution

The value of research is that the problem of offloading from power-constrained devices has been dealt with by several different types of resource management techniques. One open field research area was the decision-making of offloading based on the trade-off of energy from edge computing edge devices versus the delay if the process is offloaded, a priority-based system for workloads will be used for trade off. We can use historical analysis of different workloads on the edge computing system. Contribution made will be for high demanding QoS applications when delay of offloading is being traded off with the power that is consumed at edge device, But when the power is below a certain threshold this power consumption can be traded off with delay of offloading and now completing the task become a more priority thing over latency. The literature review done lets us know the different findings in the field of offloading in edge, fog and cloud computing taking in mind the parameters of power consumption alongside a survey on edge computing. Upcoming sections has related work, methodology, system specifications, implementation and evaluation with a ending by conclusion and future work.

2 Related Work

This section describes multiple key components of the literature review that have been conducted such as survey in fog computing , resource management schemes in fog computing, energy constraints and power consumption in edge devices, offloading schemes, components such as QOS (quality of service) response time latency, etc related to the edge/fog computing and tools for simulation which i will be using too And I did the analysis in edge computing field having devices which are resource-constrained and know about existing schemes and their results. For Example in paper like ROUTER Gill et al. (2019), talks about a approach being novel for fog enabled environment.

2.1 Survey on Fog Computing

The Fog computing survey Lan et al. (2019) tells about the principles, state-of-the-art and pending research challenges, also offers an perspective on fog computing RnD (Research and Development) and encourages further work in this field. With new paradigm shifting demands of ultra low latency, the reliability and availability due to increase in edge applications and sensors count which are very data intensive and aggressive the central clouds have become insufficient. applications examples are AI, AR, VR , Big data, Machine learning etc. Also if we keep QoS in mind then the SLA will fail with this high demanding scenario and thus we introduced fog/edge or locale computing to play. Different architectures and schemes are talked about in the literature review of the paper. Mouradian in his paper categorise the fog computing as two types firstly application specific architecture and secondly end user application agnostic. A three tier fog architecture scheme and Fog Radio access networks are mentioned by Mukherjee in the paper. Similiarly many more are there in the literature.

Fog servers are a mix, a heterogeneous mixture of devices in comparison to dedicated data servers. And scalability is there in terms millions of IoT can join in the network. Local user awareness is there so locale computing principle is met. Mobility may lead to unreliable access sometimes as the movement in wireless networks can be unreliable. Openness is a plus point in fog can support a variety of customers with various use cases and is evident by OpenFog architecture example. Requirement in architectural viewpoint and application viewpoint is discussed in the survey, also some fog programming frameworks are discussed like MobileFog , Distributed Dataflow , Hybrid cloud/Fog , FogFlow , AzureIoTEdge etc.

In the survey Open Research challenges pointed are : Proactive vs Reactive service migration in fog and how it can be done keeping in mind the QOS. Fog Infrastructure virtualisation is difficult due to heterogeneity of the fog layer and is an open research area. Resource aware dynamic reconfiguration like using of resources concerning the workloads and suggests peer to peer resource management so the big heterogeneous cluster of resources is properly utilised optimally. AI (artificial intelligence) coordination among cloud and fog is also open research area. P2P fog computing and blockchain technology as existing fog models runs on client server architecture. So Author investigated state of the art in fog programming frameworks in both academics and industrial area.

2.2 Resource Management in Edge and Fog computing

This sections deals with multiple Resource management (RM) schemes that are already existing in IoT devices with edge/fog computing paradigm with some unresolved issues.

One of the earliest problem Workload allocation problem Deng et al. (2015) in research is there which studies the trade off in offloading delay versus energy consumed in device. A proximal algorithm to reduce carbon footprint was proposed by Do et al. (2015) adjoining joint resource allocation for geographically distributed environment and it actually reduces carbon footprint while serving video streaming in a cloud server as a service. Another proposal Lee et al. (2016) was of a gateway based fog computing architecture using wireless edge devices and which manages resources usinf master-slave method. In 2017 a VRP(Virtualisation based resource provisioning) algorithm was proposed which uses distributed and parallel load balancing by author Yu et al. (2017). Similarly more RM schemes by Stojkoska and Trivodaliev (2017) Zhang et al. (2017) are proposed for managing the workloads in edge/fog computing. We can still find over provisioning and under-provisioning of resources is there in these schemes.

ROUTER : A new research Gill et al. (2019) which is validated using a simulator tool with positive results shows a novel approach and is using PSO (particle swan optimisation). While ensuring QoS (Quality of service) it lowers response time and latency as well and shows a drop in 12% in energy use in edge layer. Components of research are 1. Requirements and design of architecture in fog for applying the resource management scheme and 2. A request handler mechanism and finally 3. A study based in IoT using the simulators. An over all comparison of ROUTER over previous mentioned schemes is shown in figure 1.

Table 1
Comparison of Existing Techniques with Proposed Technique (ROUTER).

Authors	Applicable Network	Fog Nodes	Nodal collaboration	Focus	Performance Parameters (QoS)			
					Response Time	Energy	Latency	Network Bandwidth
Deng et al., 2015	Mobile Network	Servers	Master slave	Application management	✖	✔	✔	✖
Do et al., 2015	Vehicular Network	Servers	Peer to Peer	Application management	✖	✔	✖	✖
Gu et al., 2015	Mobile Network	Base Stations	Peer to Peer	Network Management	✖	✔	✖	✖
Lee et al., 2016	IoT	Network Devices	Peer to Peer	Resource Management	✔	✖	✖	✖
Yu et al., 2017	IoT	Network Devices	Peer to Peer	Resource Management	✖	✔	✖	✖
Stojkoska and Trivodaliev, 2017	Mobile Network	Base Stations	Cluster	Application management	✖	✔	✖	✖
Zhang et al., 2017	Vehicular Network	Servers	Master slave	Network Management	✔	✖	✖	✖
ROUTER (Proposed)	**IoT**	**Network Devices and Servers**	**Peer to Peer**	**Application, Network and Resource Management**	✔	✔	✔	✔

Figure 1: Comparison of different Resource management schemes over time

URMILA : Which is short for Ubiquitous Resource Management for Interference and Latency-Aware services Shekhar et al. (2020).It helps in selecting a route for offloading the workload to a upper cloud layer by making a trade off decision between edge layer and fog layer resources keeping in mind that the edge device is movable in nature. The

approach is totally dynamic as the situation of the movable device is too. it finally helps in minimising the latency without degrading the QoS. Parameters like drain of battery in the fluctuating network environment due to movements and also by the communication overhead. Distributed WAP (wireless access points) are used in experiments over cellular technology. Author comes to a optimisation problem from results of the experiment in which decision making is done for each user for where to deploy and execute the application so that minimisation of cost is achieved. This route computation is the USP of this novel approach and helps in making a informed choice for offloading.

Resource Management using LCS system : LCS is Learning Classifier system which is used by author Abbasi et al. (2020) in the research to talk about amount and range of processing workloads residing in the edge devices and how to manage a balance of workloads between this edge and the cloud. Two XCS methods are used of LCS sytems mainly XCS and BCM-XCS. Results favor for BCM-XCS as a better option to tackle this trade off decision. The system resukts shows reduction of 42% in processing delays, and is 18% more helpful in recharging the batteries than the state of the art. State of the art algorithms only take care of either cost or enrgy consumption one at a time. Proposed system is having four entities as measurable which are delay, workloads, battery status and power consumption. Delays are offloading and processing delays, and also power consumption are either computing power and operating power. Threshold of battery is used for making offloading decisions.

We an see in figure 2 the different modes of battery status of device and what decision is being made in those conditions.

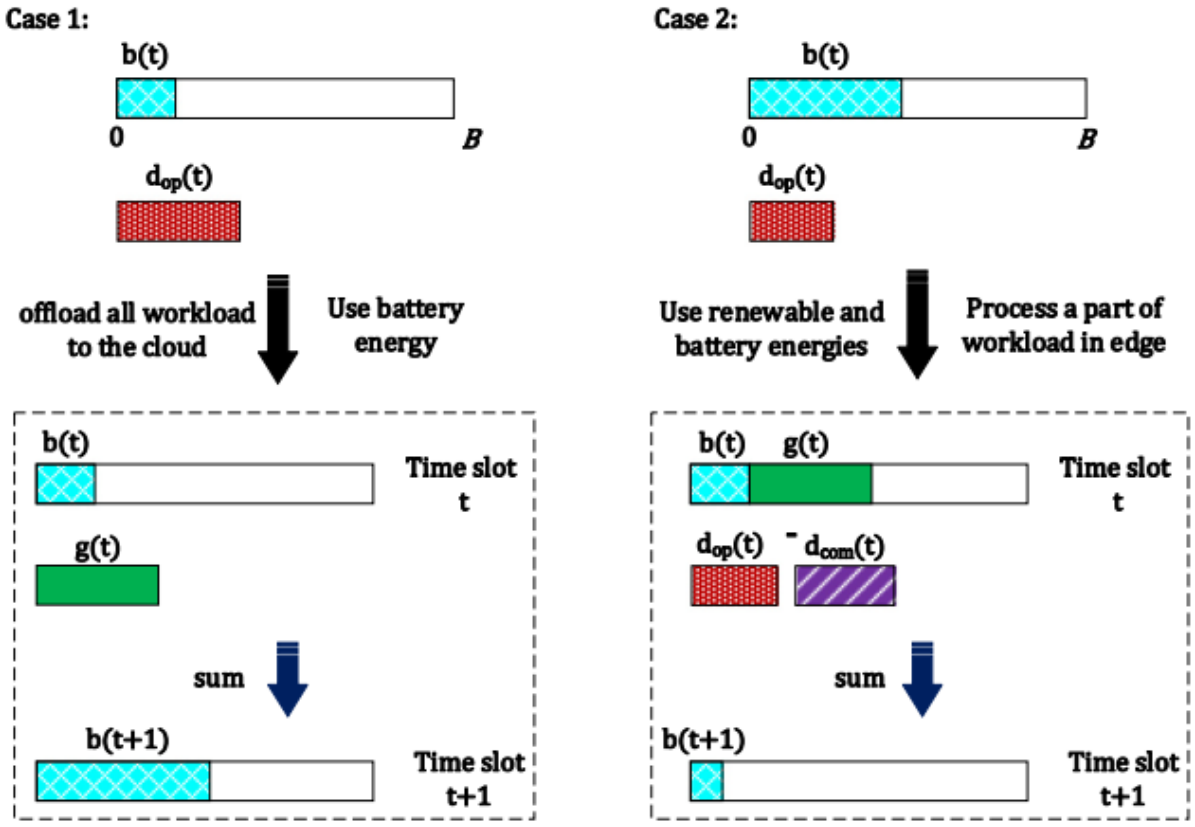

Figure 2:

Improved PSO (Particle Swan Optimization) offloading in Mobile edge computing (MEC)

Author Wu et al. (2019) talks about an improved version of the PSO algorithm that considers both energy cost and latency while offloading the workload to the cloud. The standard PSO is used and the premature defect is dealt here. System is specifically optimised as a single user system. Task division here is unique and is of two types mainly as course grain and fine grain. I am also using this one user system with multiple workloads at hand at a time. Multi user system is a future scope of the research. Literature talks about going from cloud foraging to optimal offloading of workloads and how one can also ensures QoS simultaneously. Mathematically used execution time and energy for scenarios.

E = T*P

where T is execution time, E is the total energy used and P is working power of CPU. The research talks about Improving the PSO by not falling into the local optimum and getting a global optimal solution instead which is the base for adaptive PSO algorithm

2.3 Latest Energy consumption VS Latency trade off papers

In paper Mukherjee et al. (2020) End-users in fog computing can offload the computation-intensive tasks to the nearby fog node. In addition, these tasks are often offloaded to the cloud and neighboring fog node by the fog nodes to search for additional computational tools. Author proposes a strategy in offloading to reduce the cost which is a sum of energy consumption and total delay for task processing in fog computing.

Another paper Yang et al. (2020) focuses mainly on the maritime IOT devices which does the exploration of ocean. It faces the same problem of intense data and storage and computation at the edge device and the dilemma of transmission delay versus power consumption at the edge IoT device is there. A offloading optimisation problem is formed and author come up with a proposed solution and prove using a simulation here.

In paper Lu et al. (2020) "Energy-efficient traffic offloading" a type of Heterogeneous wireless networks is purposed and shown as a efficient network structure which can handle heavy explosive data traffic. A lot of researches have talked about traffic offloading schemes, but only few talk about the energy efficiency and user mobility in this heterogeneous wireless networks for large scale. So the paper purposes a two tier heterogeneous wireless networks is purposed and it uses stochastic approaches firsthand. With the deadline and trace in place after the transfer data has been given, what hasbeen explained is a Stochastic Programming based Handover Scheduling (SP-HS). An algorithm is designed to reduce total use of energy by mobile user equipment, and this algorithm determines the change or switch taking place among networks with a movement of travelling of the device trajectory. The results which are simulated shows a positive change of 34 percent of energy could be saved by using aforementioned algorithm SP-HS.

2.4 Offloading

Offloading in cloud computing terms means when a workload or a piece of code is being taken to another place than the place of origin or the intended place to process itself. paper mentions before talks about offloading and the types as well. Also we need to avoid abuse of this offloading technique and thus avoid cyber-foraging i.e. always offloading the tasks.

Delays, Power consumption and QOS in Offloading So when we do offload we may faces some types of delays Abbasi et al. (2020) like transmission delays arriving due to network issues or size of data, or we may face processing delays due to large computational loads. Offloading can be also either offloading at the same layer or to another computational layer like edge to edge offloading or edge to fog offloading.
Different delay types we can find in paper Gill et al. (2019) while offloading a edge/mobile device workload.
Delay between edge to Fog
Delay between edge to Cloud
Network Delay:
Processing Delays
Power consumption : Whether the workload is offloaded or is run locally in the original source mobile/IoT device it will consume power. Let us assume if a workload is small and is consuming more power for offloading where it establish connection and network transmission etc is there rather than computing this small workload locally in the edge device without offloading, so it does not make sense to offload in this case and is not worth it. Paper Wu et al. (2019) explain its mathematical repercussions considering all energy consumption scenarios and project costs of offloading versus costs of running a computational workload locally, thus helping in making a informed choice of offloading and coming up with a optimal solution. But Suppose if we face a battery crises at the edge device than we have to offload most of the times Shekhar et al. (2020), so this power constraint component now matters differently depending on its level of availability in any resource/power constrained device. Different workloads depending on their size and priority like AI,machine learning, VR etc are different in terms of QoS and priority. Heavy workloads may be preferebly offloaded and hence can tolerate latency more than other apps like real time applications or health or medical related applications
Paper on offloading Lin et al. (2019) talks about mainly computational ofloading froma mobile edge computing environment. Points like task allocation, application partitioning, scheme to manage and the allocation/distribution of partitioned tasks. "However, the latency is the Achilles heel of cloud computing" so we can't ignore this latency in computational offloading.

QoS (Quality of service) is a big part of any SLA (service level agreement) document. Parameters like successful job completion, less latency play major role in forming a QoS of any SLA, i.e. the app should be fast and if not that it should at the minimum not fail. A success which is slow, with latency is preferred over a fast failure or termination of any service due to unavailable resources.
Offloading where and when The evolution in terms of computational offloading is discussed in length in the paper Lin et al. (2019). From "Odyssey back in the 1990s to the pervasive computing in 2001 (Satyanarayanan) to the amazon Ec2 in mid-2000s and cloudlet in 2009 to the MEC concept in 2014" offloading has evolved. And latest developments like 5g technologies has further reduced latency and making offloading a more viable option.

As you can see in figure 3 the architecture for edge computing with the computational offloading flow.
offloading : It happens when
Tlocal greater than Toffloading, and Toffloading = Tcomm + Tremote. (Time of offloading equals time in communicating to cloud plus time of computation in the cloud)

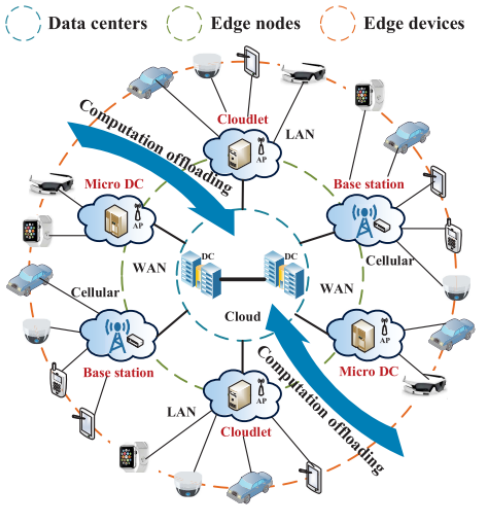

Fig. 3. *Architecture of edge computing with the flow of computation offloading. It is a typical three-tier architecture consisting of edge devices, edge nodes, and the cloud.*

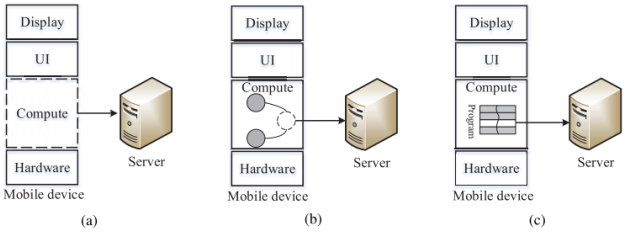

Fig. 4. *Granularity of computation offloading. Generally, mobile devices can offload computation at (a) full offloading, (b) task/component, and (c) method/thread levels.*

Figure 3:

and Elocal greater than Ecomm (Consumption of energy local greater than energy in offloading transmission in the communication).
Granularity in offloading: Full offloading or Task offloading or method/thread offloading (see figure 3).
The choosing of fog as MEC versus the centralised cloud layer for offloading is discussed in the paper Lin et al. (2019). Another solution in form a middle-ware or a cloud-let layer between these two traditional Edge and cloud layer is a cost efficient fog setup layer and is made up of low-cost single board computer clusters architecture which is proposed by author Fernández-Cerero et al. (2019).

2.5 Simulation tool used

The simulator used by me is EdgeCloudSim Sonmez et al. (2017). We will be replicating the 3 tier architecture of Edge , Fog and Enterprise big cloud servers. We will be simulating the scenarios the environment in it. Devices numbers core of processor in device and lot more technical counterparts will be defined in the configuration files of the simulator. May have to code in java functions to inherit the factory functions of the simulator as well. Will apply the proposed algorithms in the simulator and compare the analysis with other set of results without the use of algorithm and proposed environment.

2.6 Conclusion

In conclusion we can see in figure 1 from paper ROUTER as well a comparison of multiple RM schemes for offloading taking in mind the response time, energy and latency factors. This along with other literature gives us a base for taking parameters and what what are the open area of research which can be looked upon.

3 Methodology

The methodology proposed is detailed in this section and also the assumptions, specifications, architecture and system requirements around that. Latest paper Abbasi et al. (2020) helps in the decision for the energy aware trade off I am planning to do.

3.1 Proposal

The literature implies that the current scenario of edge computing doesn't support decision making of computational offloading in a single user system based on the trade off of latency/delay incurred due to offloading versus power consumption in the edge device. A new approach close to it is in recent paper Abbasi et al. (2020) using a LCS classifier system. I proposed a novel approach which handles this scenario. My resource management scheme is named as RMDPP (resource manager for delay or power using Priority-Based-Queueing). It handles the offloading of the workloads belonging to a power constrained device from a lower cloud layer (edge computing layer) to same edge layer different device or to a upper layer i.e. main cloud layer like AWS etc.

I.e. the Novel approach have to assign priority to computational workloads also later we can add feature to manual alter this priority in the system as a future work. Usually we have priority or some sort of queue assigned to these processes in the system that they are a part of. We initially use it as the initial or default priority queue for us.

3.2 System Assumptions

Many assumptions which are made to carry the experiment and the proposed approach/scheme are explained below:

One user system : For simplicity we are assuming the system to be of one user system i.e. we use one device which we will apply our solution at a time, although workloads can be multiple and varying in nature here in the mobile/IoT device.

Fallout Problem: We are assuming that energy used to offload the workload is very very low compared to the energy used to compute the workload locally. So offloading is possible at low battery levels too while on the other hand it will be difficult to compute the same workload locally. but if the energy used in offloading is more than computing it locally than offloading is not a useful solution at any cost and is the fallout problem for my proposed solution

Workload : It is a piece of code that can run independently on any platform, can be offloaded to different computing platforms and is prtioned to the lowest level possible. we have multiple workloads of different priority at a given point of time in the assumed edge/IoT/mobile system.

Weight of workloads : Weight of workload is usually directly proportional to the resource required by it for example length of task , memory required, number of cores required by workload etc. To calculate cost of offloading there are multiple methods to calculate weight of workloads and distribute it in papers like Router Gill et al. (2019)

using PSO algorithm, Urmila Shekhar et al. (2020), Also in paper Abbasi et al. (2020) which uses LCS system for weight of workloads using machine learning.

If the priority is same through the priority marker algorithm we take the weight into consideration and give high priority as a sub priority to the workload with less weight.

3.3 Partition of approach by battery threshold

System works in two partitions approach. One when battery of the system is above a certain threshold and another when this battery level is below the same threshold. We can calculate this battery threshold by the total battery required by all current workloads of the system at any given point of time, so that it is a variable and depends on the workloads present in the system. Also if the battery is on charge of the IoT/mobile device we neglect the battery percentage in that case and consider that battery is above the minimum required threshold.

4 Design Specification

4.1 System Architecture

The figure 4 shows different levels of the architecture where we have Edge layer having more parallel level devices as other IoT devices and other two layers as Fog and centralised cloud layer. It shows how the offloading is fastest to slowest from lower layer i.e. edge layer to the top layer i.e. the main cloud layer. This is the data processing stack on different layers herein.

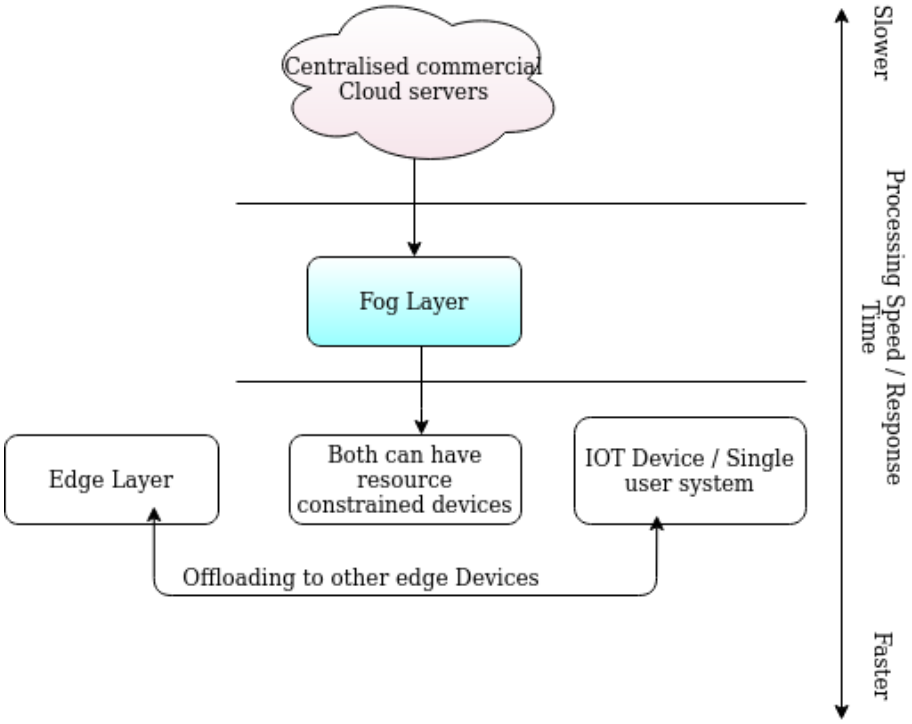

Figure 4: Data processing Layer with Response time

4.1.1 Existing Problems

Some problem we can point out in the mentioned layers of data flow are :
*** Latency** : It is a more as we travel from lower Edge layer to top layer in terms of offloading, could be either processing delays , network delays and more.
*** Power constraint** : The data collection point are mainly the IoT seneors which resides at the edge layer here, but they face a problem of resources and it could be shortage in terms of computational, power or storage. We are minly concerned with the limited power constraint of the devices at edge. For example like smartphones , smartwatches movable sensors etc. They can get critically low in terms of battery/power and can't process the computational load at hand. Our RM scheme will pick workloads from this one device and offload it parallel to another edge device or to upper cloud layers.

4.1.2 Different parts/entities for Resource management

*** App Monitor** : To get status of all existing applications we have a app monitor system like to get weight of applications, their predicted run-time, existing priority etc, theoretically it works as a dashboard for all applications with their details.

*** Historical Analysis** : From the application name we can get its historical analysis which we use to predict its run-time, priority and other empirical factors associated with the applications, right now I am using configuration in the simulator for that. For future use we can use machine learning to predict more needs of workloads and I am leaving it as a future research direction.

*** Resource Provisioning** : We are using the simulator configurations now for providing or provisioning resources to a application computational workload.

*** Priority Marker** : The second algorithm we are using is priority marker algorithm, this scheme will mark the workloads with priority based on multiple factors. It assumes that already defined priority is correct and takes that into account, and if prority is missing it will assigns one to it, also in case of same priority it assigns a sub priority depending on the weight of the workload as discusse in algorithm later.

*** Threshold** : As explained earlier it is dynamic in nature and is calculated at any point of time in the given system. In later sections it is explained properly and conditions pertaining to above and below threshold of the system
Figure 5 shows the system architecture.

4.2 System working flow

For a calculated threshold and different battery levels of a edge device or our simulated system we have different conditions when this battery level is above (or equal to) or is below then the calculated threshold. The working flow for these two scenarios are different in the proposed system. The flow of dealing with this is shown in figure 6 for different power levels as compared to the calculated threshold in the edge device. How the RM scheme will act differently in these two conditions is shown as well. The destination layer to offload the workload could be he same edge layer but to a different device or to a upper

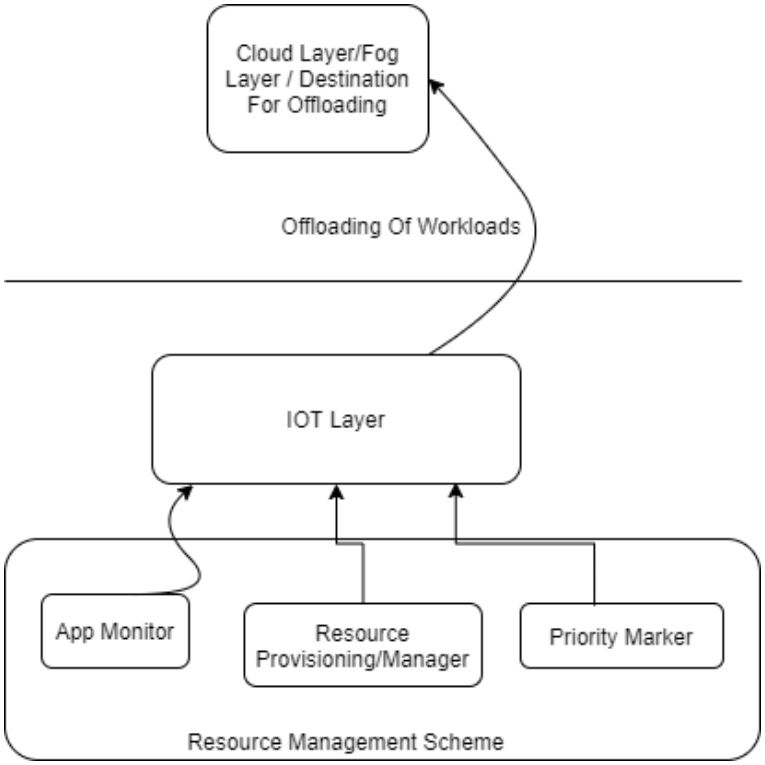

Figure 5: Offloading Scheme with Resource manager

layer of fog or centralised cloud computing.

4.2.1 Flow in case of above threshold

When battery or power levels of a system are above or equal to the threshold or even when the device is on charge we consider it to be above threshold in that particular case.

Will have normal existing probable offloading with existing algorithms, or we could use RMDPP created by me as a suggested algorithm, in which it works on the principle that high priority workload needs less latency thus will not be offloaded and less priority workloads will be offloaded for a optimal solution. So here we achieve high QoS for high priority workloads or even if we have real time workloads we prefer to compute them locally for less latency. Now suppose here we face same priority problem and suppose 3 workloads have same priority we use priority marker scheme here and mark them for sub priority.

4.2.2 Flow in case of below threshold

Now only scenario left is when device not on charge and is below a certain calculated threshold too. Now Survivability of all applications or workloads is something we have to keep in mind. We now cannot process locally at all because of the assumption that computing locally takes way more resources than offloading the same workload, and nowwe also take the priority into consideration in a different manner. Everything will be offloaded now and the high priority workloads will be offloaded first. So now the high priority workloads can compromise on latency and need to at least complete there execution.

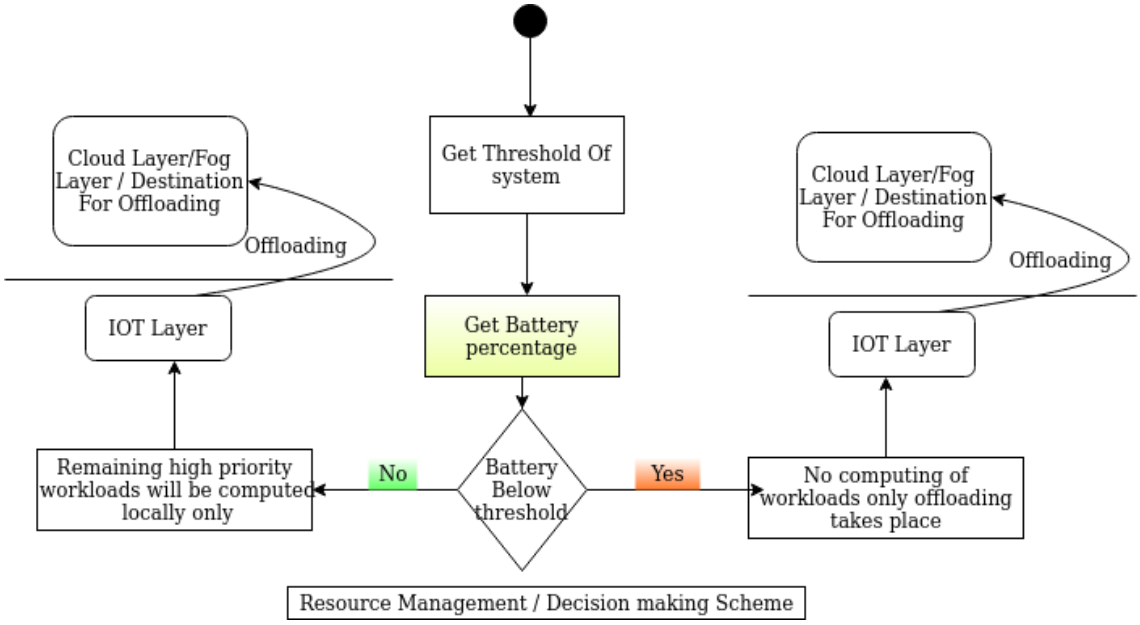

Figure 6: Types of Offloading with different battery levels

We will also see these below points happening :
* RMDPP algorithm will kick in.
* Parallel processing of workloads might stop depending on configuration.
* And now the choice will be clear to choose delay caused by latency versus the power consumed at the locale/edge device.

Fallout Problem Explained : Normally if IoT system run a workload locally it will take less time but more total battery usage. So when we offload and face latency, that is it takes more time to complete, but the energy used to unload would be lower than the energy used to compute locally. but if it is not the case then the offloading is not a viable option and I call offloading the Fallout problem. There are some methods in paper Shekhar et al. (2020) by which we can predict this so that this doesn't happen in our system and if it does we will not apply offloading to that particular workload and call it a **unconventional load** which we can't offload reasonably.

5 Implementation

5.1 RMDPP algorithm

The algorithm RMDPP(Resource management for Delay vs power using Priority-Based-Queueing) is showng the details of the steps to be followed in detail while taking the decision of offloading under different circumstances.
Let notations used in the algorithm be :
Edge device workloads be : N_{Load}
Selected computational workload in single user system be : N_i
Weight of the computational workload in edge device : W_{Load}
Battery status currently : B_{Charge}
Battery charging status (is boolean) : $B_{Charging}$

Battery threshold value : $B_{Threshold}$.

Algorithm 1: RMDPP Algorithm

Input: N_{Load}, W_{Load}, B_{Charge}, N_i, $B_{Threshold}$;
Output: Selected workloads to offload;
Result: Queue of workloads with priority marking to offload
initialization : Workloads which are sorted by priority will pick from the top ;
If selected workload don't have priority call proirity marker algorithm ;
If priority exists call the priority marker algorithm ;
while *each* N_i *is processed till than* **do**
    Case 1;
    if $B_{Charge} > B_{Threshold}$ *or* $B_{Charging}$ **then**
        Step 1 : execute the higher priority workloads locally for less latency sorted by higher priority to lower priority based on the priority based queue ;
        Step 2 : Also offload the lower priority workloads to cloud from the priority queue , maybe can take the lower half of the queue and start with high to lower priority ;
    else
        Case 2;
        if $B_{Charge} <= B_{Threshold}$ *or* $B_{Charging}$ **then**
            Step 1 : Since battery status is too low now so offload as much as possible and start by high priority to low priority from the priority queue so there QoS is met by completing the high priority wotrkloads first as the device battery is too low to do anything. ;
        end
    end
end

Complexity : Complexity will in linear range for n elements as we have n elements queue and we are taking one item at a time from the queue and final destination is one only here as edge layer or cloud layer depending on the configuration.

5.2 Priority based division

We need to have a priority when choosing workloads over a pool of multiple workloads from the device in experiment. For now we use a priority marker algorithm to assign missing priority if any in the system full of applications/workloads and is explained in later sections.

Priority marker is the second algorithm we are using it is being called from the first algorithm and runs independently too, It shows how we create or decide priority and how we can compare in a n items system and also in the existing priority marker items or system.

Algorithm 2: Priority Marker Algorithm

Input: N_{Load}, W_{Load}, N_i, Using quick sort and priority and we get sorted list;
Output: For RMDPP use a queue of sorted workloads with priority;
Result: for queueing a sorted list of workloads
initialization : Will pick a workload or receive a item to prioritise;
while *in the priority queue each* N_i *is Sorted* **do**
    Case 1;
    if *Existing priority of* N_i > *Existing priority of* N_{i+1} **then**
        Step 1 : in the fifo queue Put N_i ;
    else
        Case 2;
        if *Existing priority of* N_i = *Existing priority of* N_{i+1} **then**
            Step 1 : Use W_{Load} of N_i and N_{i+1} and give higher priority to the one with less weight ;
            Step 2 : Insert N_i in the fifo queue using the priority gotten by weight ;
        end
    end
end

Complexity: To calculate complexity of this priority marker algorithm is the same as quick sort O(log(n)) (or any other recommended sorting algorithm) + O(n)
here n is total workloads, and n because we have to pick 2 items from already sorted list and put one to the queue if same priority is there we use weight as a another parameter to determine higher priority from the two, each time we are picking 2 elements so complexity should be O(n) for this particular sub-part of the algorithm.

5.3 Threshold explained

Battery above threshold : Usually in this scenario the high priority computational loads will be the one which need less latency as per QoS i.e. if computing locally is faster it will compute locally only and thus latency sensitivity will be taken into consideration here.
Battery below threshold : Now the battery is below a certain threshold and as per system assumptions workloads now are difficult to compute locally and offloading can be done in this case for all workloads and is feasible i.e. the fallout problem is not in effect. Now we use the priority of workloads and the RMDPP algorithm to offload the high priority workloads first, but they are ok with this latency which wil be there while offloading. As now the need to complete the tasks overthrow the latency sensitivity and QoS for completion of task is of highest priority now for the same task.

"This Novel approach protects the benefit of high priority workloads and their QOS(quality of service) too till the time it can, and does all this using trade off between the power consumed versus the delay faced by the computational workload."

5.4 Using Simulations and code to implement

EdgeCloudSim : is a simulator which provides a environment to simulate Edge computing scenarios where it is possible to conduct experiments that considers both com-

putational and networking resources. EdgeCloudSim is based on CloudSim but adds considerable functionality to it. Also as a developer we can inherit and code it's factory classes and develop the classes and entities as per needs and scenarios.
Eclipse IDE : Eclipse a java IDE (integrated development Environment) is used to code or modify the Edgecloudsim Java related code and to manage the configurations to run the simulations over the application. What I have tried is Run simulations to prove the above algorithms in the project. I have used EdgeCloudSim a Java based simulator which can be modified by coding in java as per your need. After Applying the Priority marker algorithm code in the java code base priority for critical applications as high priority applications.
GNU Octave: It is a open source tool which is used to run files of Matlab extensions i.e. .m files. I was using Matlab code to generate graphs of raw data genearted from very heavy simulations run over long period of time for many parallel processes with multiple iterations over a big number of edge devices ad some simulated cloud or Fog servers, so the count of logs or results from simulations was too big and to consolidate the results Matlab scripting code was a good choice. But GNU was open source and a free tool to run these Matlab scripts so it was used her.

6 Evaluation

To run Multiple scenarios I had to code and had to change to many configurations combinations to apply the algorithms. Code was done for queuing in java, also java code I created scenarios as used in below experiments. Also coded in python to call multiple iterations of a java file with passed parameters too.
WE are using four types of applications :
1. AUGMENTED REALITY
2. HEALTH APP
3. HEAVY COMPUTATIONAL APP
4. INFOTAINMENT APP

6.1 Running processes on a default simulations environment

Before coding a default simulation scenario was run with taking mentioned applications:
We can see that all apps have a probability of offloading, but HEAVY COMPUTATIONAL APP is most needed to offload as it is heavy in nature and it will be logical to offload this in normal conditions so that we can use a high resource cloud environment to run the process faster.
Now also the highest priority to complete will go to health app in any case.

We did run simulations with these applications with some decided configurations where we have a 3 tier architecture of cloud , Fog/Edge computing and tier 1 as mobile devices. We can observe that failure percentage and processing time of apps in Tier 1 i.e. mobile devices layer is higher than Edge/Fog or actual cloud tier. So keeping all applications altogether in tier 1 pose a threat to more failure rates of applications and thus SLA of applications is not met always if the offloading and selection of applications is not done properly. We can see and analyse results in figure 7 below :

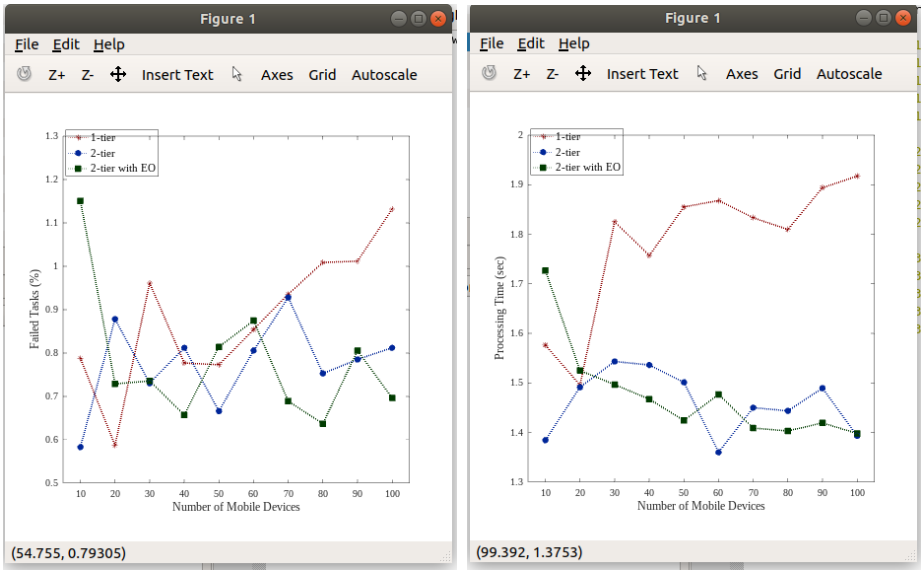

Figure 7: Failed tasks AND Average Processing time over different tiers

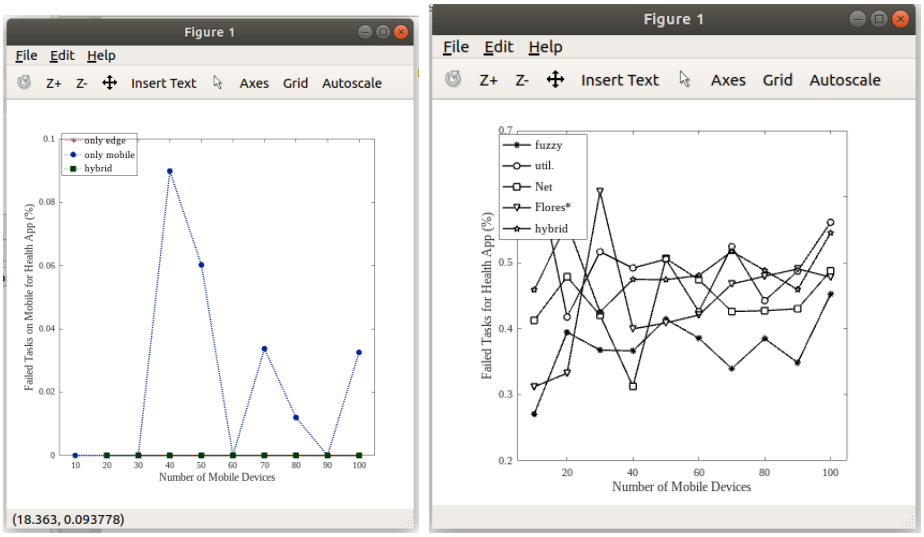

Figure 8: Failed health tasks over mobile and over different algorithms

6.2 Case of battery constraint using Priority algorithm

In this Case we have a constraint of battery/resource to run the applications and we will see its effect on different applications completion or failure rate. We use different offloading algorithms/logic to see if they effect the high priority failure rates or they don't help in it at all.

We see that failure rate of a particular task type is not helped by these algorithms. We had marked some apps such as health application as a high priority app with using Java code in the code base, but to effectively use the priority we have to use the RMDPP algorithm with it. We can see and analyse results in figure 8 above :

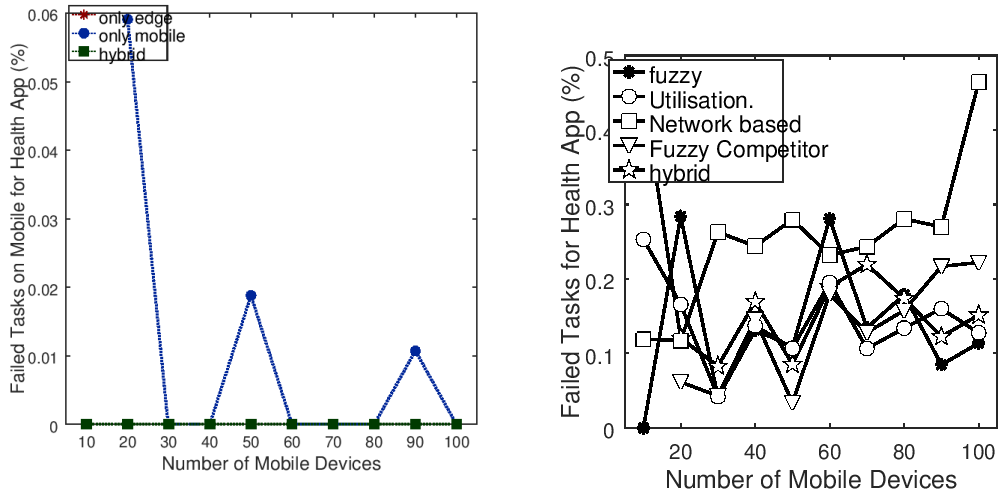

Figure 9: Failed health tasks over mobile and over different algorithms with RMDPP and priority algorithm

6.3 Case for Running RMDPP And Priority algorithm together

Now for this scenario we use both algorithms RMDPP and priority marker algorithm. So now we have battery below the threshold calculated to run the applications.

Now the top priority application we prioritise is health app lets analyse the implications on health app before and after the application of the algorithms. We now see with different already existing offloading techniques like fuzzy , utilisation based, network based , hybrid techniques the results of these before and after applying over RM scheme we see that fail percentage of one particular high priority app is lowered here. So it helped in lowering the failure rate of the health app, also same result is promised for other apps as well. So an overall less percentage of failing of tasks is observed with the use of RMDPP and priority algorithm used with traditional offloading algorithms. Please see Figure 9 for results :

We can see reduction in failed tasks percentage for health app over mobile/edge layer using our algorithm as compared to case 2, and similarly reduction is there for failed tasks percentage for different offloading schemes used in case 2 versus the case 3 experiment which uses our algorithms.

In Table 1 results from case 3 experiment and case 2 experiment are compared. We see that network based algorithms reults are not changed much as our algorithms don't do any network related optimisations or changes.

Table 1: Case 2 versus case 3 experiment for lets say 40 edge devices in both experiments

offloading Algorithms	**case 2 fail %**	**case 3 fail % with RMDPP**
Fuzzy	0.37%	0.14%
Utilisation	0.48%	0.13%
Network based	0.30%	0.28%
Fuzzy competitor	0.40%	0.15%
Hybrid	0.48%	0.18%

6.4 Discussion

From above experimentation we have gathered some significant data of different conditions of the multivariate environment setup for offloading scheme that has been proposed alongside other conventional schemes(fuzzy, utilisation , network base, hybrid algorithms) and how the proposed scheme has promising results for proposed area of benefit only. Our proposal was to benefit high priority applications completion only in case of a battery or resource crises using offloading technique. or we can call it choosing some over few type implementation to reduce failure of high priority processes. In our case we consider health app as high priority workload than Heavy computational and so on by using priority marker algorithm, later we use RMDPP algorithm to ensure if resource or energy is below a certain threshold in edge/mobile device we can only offload applications that too in a certain order marked by priority. This reduces overall failure of critical applications as shown in results. In literature review we can see that this novel approach has shown some new findings as compared to other algorithms and we can use it on a conditional base depending on use case need.

7 Conclusion and Future Work

Paper proposes to achieve a trade off decision making when it comes to offload to cloud versus running a process locally so that we can achieve maximum completion of workloads based on priority as a logical inference of the question. And we offered a novel approach having two algorithms to do so. Although our whole solution is conditional based i.e. when the battery/resource constraint arises the devices face the dilemma of choosing between offloading the task or to run it locally and which order to follow for offloading only and stop locale computing. I did experiments to prove this point of failure of priority apps over non priority apps, and experiment to prove use of our novel approach to give high importance to high priority applications for completion. I got results as explained above for a betterment of high priority apps over low priority and also the dilemma trade off decision of power consumption vs offloading delay is now dealt with. For example health app after a high priority shows less failure percentage as shown in table 1 above.

For future work we can use machine learning to automatically decide this priority marking of applications based on historical data and analysis. And also what would be the best order for offloading across edge or multiple mobile devices coming into play for this scenario of priority offloading/ completion of applications over full Edge network. Also an open research area could be suggesting this cost-effective fog layer Fernández-Cerero et al. (2019) using already existing state of the art resource management schemes and replacing the cost of fog from this new cost and thus make a new decision based on change results. Also in paper Abbasi et al. (2020) calculation of weight of workloads can help further in this decision making and lead to a better and precise system for offloading.

References

Abbasi, M., Yaghoobikia, M., Rafiee, M., Jolfaei, A. and Khosravi, M. R. (2020). Efficient resource management and workload allocation in fog–cloud computing paradigm in iot using learning classifier systems, *Computer Communications* **153**: 217 – 228. Impact

Factor: 2.613.
URL: *http://www.sciencedirect.com/science/article/pii/S0140366419318274*

Cui, K., Sun, W. and Sun, W. (2019). Joint computation offloading and resource management for usvs cluster of fog-cloud computing architecture, *2019 IEEE International Conference on Smart Internet of Things (SmartIoT)*, pp. 92–99.

Deng, R., Lu, R., Lai, C. and Luan, T. H. (2015). Towards power consumption-delay tradeoff by workload allocation in cloud-fog computing, *2015 IEEE International Conference on Communications (ICC)*, pp. 3909–3914. Core Ranking: B.

Do, C. T., Tran, N. H., Chuan Pham, Alam, M. G. R., Jae Hyeok Son and Hong, C. S. (2015). A proximal algorithm for joint resource allocation and minimizing carbon footprint in geo-distributed fog computing, *2015 International Conference on Information Networking (ICOIN)*, pp. 324–329. Core Ranking: B.

Fernández-Cerero, D. (1, . ., Álvarez García, J. . . ., Soria-Morillo, L. . . ., Fernández-Montes, A. . . . and Fernández-Rodríguez, J. . . . (2019). Single-board-computer clusters for cloudlet computing in internet of things., *Sensors (Switzerland)* **19**(13). Impact Factor: 2.475.
URL: *http://search.ebscohost.com/login.aspx?direct=trueAuthType=ip,cookie,shibdb=edselcAN=eds52.0-85070184096site=eds-livescope=sitecustid=ncirlib*

Gill, S. S., Garraghan, P. and Buyya, R. (2019). Router: Fog enabled cloud based intelligent resource management approach for smart home iot devices, *Journal of Systems and Software* **154**: 125 – 138. Impact Factor: 2.559.
URL: *http://www.sciencedirect.com/science/article/pii/S0164121219300986*

Karamoozian, A., Hafid, A. and Aboulhamid, E. M. (2019). On the fog-cloud cooperation: How fog computing can address latency concerns of iot applications, *2019 Fourth International Conference on Fog and Mobile Edge Computing (FMEC)*, pp. 166–172.

Lan, D., Taherkordi, A., Eliassen, F. and Horn, G. (2019). A survey on fog programming: Concepts, state-of-the-art, and research challenges, *Proceedings of the 2nd International Workshop on Distributed Fog Services Design*, DFSD '19, Association for Computing Machinery, New York, NY, USA, p. 1–6.
URL: *https://doi.org/10.1145/3366613.3368120*

Lee, W., Nam, K., Roh, H. and Kim, S. (2016). A gateway based fog computing architecture for wireless sensors and actuator networks, *2016 18th International Conference on Advanced Communication Technology (ICACT)*, pp. 210–213.

Lin, L., Liao, X., Jin, H. and Li, P. (2019). Computation offloading toward edge computing, *Proceedings of the IEEE* **107**(8): 1584–1607.

Lu, F., Hu, J., Yang, L. T., Tang, Z., Li, P., Shi, Z. and Jin, H. (2020). Energy-efficient traffic offloading for mobile users in two-tier heterogeneous wireless networks, *Future Generation Computer Systems* **105**: 855 – 863.
URL: *http://www.sciencedirect.com/science/article/pii/S0167739X17317740*

Mukherjee, M., Kumar, V., Kumar, S., Matamy, R., Mavromoustakis, C. X., Zhang, Q., Shojafar, M. and Mastorakis, G. (2020). Computation offloading strategy in heterogeneous fog computing with energy and delay constraints, *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, pp. 1–5.

Shekhar, S., Chhokra, A., Sun, H., Gokhale, A., Dubey, A., Koutsoukos, X. and Karsai, G. (2020). Urmila: Dynamically trading-off fog and edge resources for performance and mobility-aware iot services, *Journal of Systems Architecture* **107**: 101710. Impact Factor: 0.913.
URL: *http://www.sciencedirect.com/science/article/pii/S1383762120300047*

Sonmez, C., Ozgovde, A. and Ersoy, C. (2017). Edgecloudsim: An environment for performance evaluation of edge computing systems, *2017 Second International Conference on Fog and Mobile Edge Computing (FMEC)*, pp. 39–44.

Stojkoska, B. R. and Trivodaliev, K. (2017). Enabling internet of things for smart homes through fog computing, *2017 25th Telecommunication Forum (TELFOR)*, pp. 1–4.

Weijian, L., Yingyan, J., Yiwen, L., Yan, C. and Peng, L. (2018). Optimization method for delay and energy consumption in edge computing micro-cloud system, *2018 5th International Conference on Systems and Informatics (ICSAI)*, pp. 839–844.

Wu, J., Cao, Z., Zhang, Y. and Zhang, X. (2019). Edge-cloud collaborative computation offloading model based on improved partical swarm optimization in mec, *2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 959–962. Core Ranking: B.

Yang, T., Feng, H., Gao, S., Jiang, Z., Qin, M., Cheng, N. and Bai, L. (2020). Two-stage offloading optimization for energy–latency tradeoff with mobile edge computing in maritime internet of things, *IEEE Internet of Things Journal* **7**(7): 5954–5963.

Yu, L., Jiang, T. and Zou, Y. (2017). Fog-assisted operational cost reduction for cloud data centers, *IEEE Access* **5**: 13578–13586.

Zhang, H., Zhang, Y., Gu, Y., Niyato, D. and Han, Z. (2017). A hierarchical game framework for resource management in fog computing, *IEEE Communications Magazine* **55**(8): 52–57.