International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. #, No.# (Mon-20..)

Latency Management in Task Offloading from IoT to MEC

Eram Fatima Siddiqui[1] and Sandeep Kumar Nayak[2]

[1]Department of Computer Application, Integral University, Lucknow, India
[2]Department of Information Technology, Babasaheb Bhimrao Ambedkar University, Amethi, India
E-mail address: eramfatima24894@gmail.com, nayak.kr.sandeep@gmail.com

Abstract: Mobile Edge Computing is a state-of-art technology which is being used to providereal-time environment by computing and responding in shorter timelines to the IoT generated requests. The task computation requests being sent to these servers is called task offloading, which is a highly complex process. The decision to offload a task for remote computation is done with the aim of receiving responses within few instant but these server in turn gets heavily loaded with thousands of computation requests as each server is connected to a number of IoT devices. This may result in situations like imbalanced workloads and resource starvation. The occurrence of this situation is caused due to adoption of full task offloading policy targeted IoT environment. Many works have already been observed in order to improve this offloading approach but it remains a complex issue. In this research study it is being tried to propose a latency minimizing procedure with optimal task splitting method. This will not only prevent resource starvation but also reduce total incurring latency and lead to quick responses. The proposed work will facilitate parallel remote and local computationof task and thus reducing the total computation time with optimal set of resources. The proposed model has been validated using hypothesis testing including Shapiro-wilk, One-wayANOVA Test, F-Test two-sample Z-test, Multiple Linear Regression Test and was successfully found to be efficient in minimizing latency with the use of partial offloading policy and have resulted in optimal resource allocation when compared to other traditionally existing offloading policies.

Keywords: *Mobile Edge Computing, Internet of Things, Task Offloading, Latency, Task Splitting, Resource Allocation*

1. INTRODUCTION

The rapid advancement in the usage of mobile networks and IoT technology, has resulted in a lot of improvement seen in computation-intensive and time-critical applications such asaugmented reality, speech-recognition, speech-to-text translation and gaming. Along with this IoT technology has revolutionized all global sectors from agriculture to industrial growth and from education to travelling, everything has become digitally smart with intensive use of IoT.However, these devices are power and resource-constrained devices due to which they do not meet the increasing computational demands and provide quality of experience to end users to stringent real-time level. The power limitation and limited availability of resources to compute complex tasks is a remarkable issue to solve [1]. This is why the current network architectures are failing to deal with enormous amounts of data traffic being generated every second from hundreds of IoT nodes. The main challenge is to process this data successfully inorder to deliver value data out of it. Therefore in order to meet the latency and computation requirements, cloud has emerged to be the most promising solution. Cloud is capable of providing rich amount of resources and other computational services to these resource-intensive applications without the resource starvation situation to occur. Along with this, the approach helps in less power consumption for IoT devices by offloading these computable tasks to remote cloud centres [2]. However, the approach has many drawbacks like location- unawareness, mobility support, high latency, late response time and others which may negatively affect real-time computation need and user experiences [3]. Mobile Edge Computing technology emerged as a solution in order to combat these drawbacks. Thetechnology will not replace the cloud but will create an add-on to enhance the computing and response functionalities of the cloud for time-intensive and computation-intensive IoT applications.

E-mail:author’s email

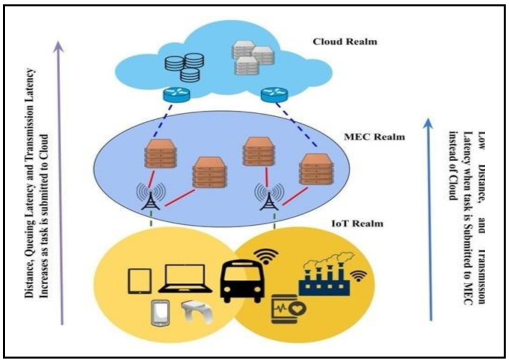
Cloud Realm
MEC Realm
IoT Realm
Distance, Queuing Latency and Transmission Latency Increases as task is submitted to Cloud
Low Distance, and Transmission Latency when task is Submitted to MEC instead of Cloud

Figure 1. Figure illustrating the need of using Multi-Access Edge Computing with Cloud Computing

Offloading the tasks for purpose of computation to near Edge servers instead of cloud providefaster responses with minimal latency and enough availability of required resources. Although, with benefits this kind of remote computation method also increases the overhead of transmitting the task towards edge server [4]. The location at which the associated edge server may be residing, also play a crucial role in adding up of the overhead. Therefore, it is very necessary to determine whether it will be an optimal solution to opt for remote computation or not. This involves comparison of total time a task would take to transmit to the server with task requirements and its deadline. From here, the final offloading decision to be taken by the mobile device should depend upon three checkpoints a) Whether to offload ornot b) To offload to edge server c) To offload to Cloud.

Another major checkpoint on which the task offloading decision depends is task computation requirements. If these requirements do not match local resource availability like non-availability of required resources, incapability of computation resources, energy constraints etc., it has to be offloaded to an MEC server for computation. This adds up to the delay including transmission delay and again waiting of task in the MEC's waiting and service queue. Also, even if the edge server does not match the task requirements, finally it has to be offloaded to cloud which doubles up the total overhead and latency.

The task offloading problem has been studied extensively and critically in cloud computing, but still the problem remains unsolved to much extent even after the introduction of MEC computing. Some major issues as discussed include added latency and late response time, energy consumption, unfair allocation of workloads and extra transmission times. These considerable challenges may affect the overall performance of the model and nullify optimal resource allocation [5]. The solution proposed in this paper to combat the task offloading problem includes that the total number of task traffic be divided and allocated for parallel remote as well as local computation so that totally occurring delay overhead can be minimized. This can be done by dividing the task into subtasks and running them locally as well as remotely on edge server. This kind of approach will not only minimize the latency butwill also improve the QoS and response requirements. However optimal task splitting decision is a challenging task and require a lot of parametric based pre-calculation and knowledge of device, edge and cloud status.

Another requirement for IoT-MEC architecture is the presence of edge server at nearest position within the network. Therefore, the distance parameter also comes into play for causing latency and optimal computed results. The association of any user-end device should be with the nearest edge server so that there is minimum occurrence of task migration, queuing and service latency with optimal use of network and resources. The proposed latency minimization IoT-MEC model is capable of minimizing multiple types of latencies detected during data transmission in uplink direction and meets the real-time scenario.

The goal of this paper is to:

- Building of nearest distance IoT-MEC Pair for minimum transmission latency.
- Design a double parallel-remote task computation model.
- Design a parametric task offloading decision algorithm for minimum latency and optimal resource utilization.

The rest of this paper has been organized as follows: Section II discusses about the previous works proposed in order to carry out optimal task offloading with minimal latency and maximum resource utilization, Section III discusses the proposed task offloading methodology with the aim to reduce total latency and provide real-time responses, Section IVdiscusses the various policies which will be used collectively in this research paper, in Section V various Hypothesis Tests have been performed in order to validate normal distribution of considered dataset and proposed task offloading algorithm, Section VI analyse the outputs or results when the proposed algorithm is compared with traditionally existing task offloading approaches, Section VII have a brief discussion of this research paper and finally Section VIII discusses the future work which may be done to enhance the proposed task offloading approach.

TABLE I. S

UMMARY OF USED NOTATIONS

N	Set of MEC

	Servers
Λ	Random Task
$tTrans_{\Lambda}^{t}$	Task Transmission Rate
α_{iC}	Server-cloud association indicator
t_{dist}	Distance between device and server
ra	Resource Allocation
t_{off}	Task Offloading decision time
Λ_{ID}	Unique Task ID
ly	Latency
T_{cmp}^{C}	Computation Time of Task at Cloud
$n_{packets}$	Total Number of Task Sub-units
d_{Λ}^{local}	Local task size
t_{off}	Task offloading time
t	Internal Decision parameter
T_{wait}^{j}	Time task has to wait at server to get serviced
$\mathrm{WL_i}$	Current workload on device
U	Set of IoT nodes
Λ_{TD}	Task Deadline
d_{Λ}	Data size of task
α_{ij}	Server-Device association indicator
c_{Λ}	Total number of CPU cycles
WL_{Λ}	Task Workload
T_{cmp}^{i}	Local Computation Time of Task
T_{cmp}^{j}	Remote Computation Time of Task
ec	Energy Consumed
$tTrans_{\Lambda}^{t}$	Total transmission time towards server
P	Pivot Point
d_{Λ}^{remote}	Remote Task Size
t_{split}	Task Splitting Time
T_{wait}^{i}	Time task has to wait at device to get serviced
$Tcmp$	Total Computation Time
$\mathrm{WL_j}$	Current workload on server
WL_{max}^{i}	Threshold to process workload locally
WL_{rem}^{local}	Remaining workload capacity of device
M_{free}	Idle Memory
WL_{max}^{j}	Threshold to process workload remotely
WL_{rem}^{remote}	Remaining workload capacity of server
M_{alloc}	Memory in use

2. RELATED WORK

Swain et al. have proposed Matching-Theory-Based Efficient Task offloading framework which aims to reduce the total system energy and remove any outage activities which do not match the strict latency constraints within an IoT-Fog network. The framework uses many-to-one matching game theory for resource allocation [6]. Ali et al. have proposed a jointapproach for minimizing latency and allocation of resources. The authors have used many-to-one matching game theory where both IoT and Fog nodes are capable of self-organizing themselves in order to solve the game [7]. Xue et al. have proposed a novel approach which uses NOMA in order to transmit those tasks which need offloading towards the Edge-servers, thereby maximizing the system's processing capability and use of resources [8]. Rafiq et al. used Khun-Munkres algorithm to give the optimal solution for cell association and computational offloading in MEC-IIoT networks [9].

Xia et al. have proposed a joint scheme for task offloading and resource allocation using Lyapunov optimization theory [10]. Senthil have developed a resource scheduling method with minimum energy consumption and optimal use of resources within strict deadlineconstraints using the SFBL method [11]. Cui et al. have proposed an MEC-SAT network based on IoT technology for minimal latency occurrence and use of total energy during data transmission, thereby doing task scheduling and resource scheduling [12]. Alameddine et al. have studied the DTOS problem and tried to solve the resource scheduling and allocation problem in an MEC enabled network [13]. Liu et al. have investigated the task splitting for parallel task computation with optimal use of resources in an Multi user-based environment [14].

Almughalles et al. have studied fog cell formulation and selection on a joint basis for optimaluse of resources and minimal latency [15]. Gu et al. have facilitated a kind of tutorial for resource management methods and resource selection [16] . Do et al. have tried to solve the convex problem of optimization for a distributed environment for the purpose of resource allocation [17]. Deng et al. have worked on the basis of fog computing environment to develop a framework for minimal energy use and task allocation with optimal use of available resources [18]. Zeng et al. have investigated the total time used or consumed for completing a randomly arrived task in an FC-SDES environment for optimal task placement and scheduling [19]. Zhang et al. have proposed a framework for multiple fog nodes, using Stackelberg game pricing problem and many-to-many matching theory [20]. Elbamby et al. have proposed a model for optimal distribution of task and proactive computing mechanism urder URLLC constraint [21]. Yang et al. have proposed a model for multi-user MECenvironments in order to minimize the task computation latency [22].

Nikaein et al. have an low latent MEC-based framework for decision coordination among distributed segments of network [23]. Brik et al. have a novel placement model for federated platform in order to provide minimal latency with maximum utilization of resources and full service availability in an MEC-based environment [24]. Alnoman et al. have proposed an SCMA-based scheme based on factors like latency, throughput and connectivity [25]. Han et al. have studied a UAV-based MEC system for the purpose of enhancing computational capability and carry out a task-splitting procedure over an optimal rate [26]. Gu et al. have proposed a novel MEC-based framework for dynamic IoT nodes based on federated intelligence [27]. Kovacevic et al. have studied the computational offloading concept under strict latency constraints with maximum usage of available resources [28].

Yoshino et al. have proposed an adaptive scheme for statistically generated data in order to control data traffic. Chen et al. have proposed a control and safety system for IoT based application with minimal latency and energy constraints [29]. Hong et al. have studied a distributed antenna system for uplink transmission in an IIoT system and environment [30]. Yang et al. proposed a scheme for uplink data transmission under the 5G communication model [31]. Ismail et al. have proposed an AGCM based model for data aggregation at cloud with maximized retransmission rate and throughput [32]. Zhang et al. have investigated the MEC as well as IoT environment for task offloading used hierarchical framework with minimum energy consumption [33]. Germenis et al. have proposed a cross layer optimization framework based upon the concepts of green computing and intelligent communication methods and minimal latency and enhanced reliability constraints [34].

Calice et al. have proposed and implemented a low cost and reliable seismograph which is configurable, low latency and reliable for early warning of earthquakes [35]. have investigated various offloading strategies with minimal energy consumption and maximum use of resources. Kherraf et al. have proposed a mathematical framework for task assignment and resource allocation [36]. Hao et al. proposed a highly reliable wireless model based upon 5G technology which satisfies the transmission and latency constraints [37]. Park et al. have presented a novel task distribution scheme with fair workload balancing based on the queue maintained by each server based on an MEC environment [38]. The following Table 1 describes an analysis of past work done on basis of latency, task offloading and optimal resource allocation challenges:

TABLE II. S

UMMARY OF USED NOTATIONS

Ref. Notation(RN_i)	**Objective**	**Parameters**		
		ly	t_{off}	ra
RN_1	To minimize total	↑	↓	↓

	energy consumption and to meet the deadline in the IoT-Fog internetwork.			
RN_2	To jointly handle the workload and latency factors in IoT-Cloudlet network within strict deadline requirements.	↑	↓	↑
RN_3	To handle task-offloading and resource optimization activities within a multi-user, multi-task and multi-server environment.	↓	↓	↑
RN_4	To investigate the cell formulation and interconnection for computational offloading in an MEC-IIoT environment.	↑	↓	↓
RN_5	To tackle and optimize the problem of heterogeneous task-offloading and resource allocation in an MEC-IoT network.	↑	↓	↑
RN_6	To effectively allocate resources to the offloaded task with minimum energy consumption and latency.	↑	↓	↑
RN_7	To optimize latency and energy-emission constraints in an MEC supported SAT-IoT network.	↑	↓	↓
RN_8	To tackle the offloading of heterogeneous tasks and its simultaneous resource allocation problem in an MEC limited capable environment.	↓	↑	↑
RN_9	To investigate a random task offloading framework in a multi-user environment with task partitioning.	↑	↓	↑
RN_{10}	To address computational offloading and cell selection problem jointly in order to minimize task computation latency.	↑	↑	↓
RN_{11}	To provide optimal solution for resource allocation by the help of matching theory.	↓	↑	↓
RN_{12}	To minimize user traffic towards centralized data center and minimize carbon footprint and resource allocation problem.	↓	↑	↓
RN_{13}	To tackle the trade-off relationship that exists between the energy and latency factors in a fog-cloud environment.	↑	↓	↑
RN_{14}	To sustainably lay task images on any embedded or server-based platform supported by fog computing environment.	↓	↑	↑

RN_{15}	To propose a general framework for all fog nodes, data service operators, data service subscribers in order to achieve the optimal resource allocation schemes in a distributed fashion.	↓	↓	↑
RN_{16}	To provide an environment where cloudlets may proactively cache task popularity with minimal latency constraint.	↑	↓	↓
RN_{17}	To address the task offloading and computational resource allocation jointly with minimum latency.	↓	↑	↑
RN_{18}	To propose a Low-Latency Multi-access Edge Computing platform thereby enabling mobile network monitoring, control, and programmability.	↑	↓	↑
RN_{19}	To address the problem of deploying MEC based applications over a federated edge infrastructure in order to meet strict latency and computational requirements.	↑	↓	↑
RN_{20}	To investigate the proposed framework under various SCMA configurations in order to verify various factors like connectivity, throughput, task completion time, and complexity.	↑	↓	↑
RN_{21}	To minimize energy consumption and resource allocation problems in a distributed MEC environment.	↓	↓	↑
RN_{22}	To reduce various occurring latencies and enhance reliability in the Edge-IoT environment.	↑	↓	↓
RN_{23}	To propose a joint solution for computational offloading and resource allocation with strict latency constraints.	↑	↓	↑
RN_{24}	To propose a scheme for aggregation of statistical data with real-time latency constraints for controlling generated data traffic.	↑	↓	↓
RN_{25}	To investigate the use of minimum power and energy transmission for uplink data transmission in a URLLC environment.	↑	↓	↓
RN_{26}	To design a distributed antenna system for complying to strict latency and reliability requirements for uplink transmission in	↑	↓	↑

	IIoT.			
RN_{27}	To propose such a scheme that could provide low latency and do resource management with a fixed period of time.	↑	↓	↑
RN_{28}	Have proposed a queue based model for green cloud with reduced energy and latency thereby reducing congestion.	↑	↓	↓
RN_{29}	To address performance measures for energy efficiency and workload offloading in an MEC environment.	↓	↑	↓
RN_{30}	To propose a scheme to wave-off the trade-off between enegy and spectrum efficiency in 6G enabled IoT network.	↑	↓	↓
RN_{31}	To design a low cost and smart and configurable seismograph, that is capable of supporting seismological and geophysical data in real-time	↑	↓	↓
RN_{32}	To provide a model for task splitting for parallel local and remote computation of subtasks within the same task and same time.	↑	↓	↑
RN_{33}	To study the workload assignment within the latency and reliability constraint.	↓	↑	↓
RN_{34}	To implement IoT-Grid(IoT-G) structure for successful broadband transmission within the existing bandwidth.	↑	↓	↓
RN_{35}	To enhance efficiency of available resources in a multi-access MEC environment with fair workload distribution.	↑	↑	↑
Propose Work	**To provide task computation with minimal latency, energy and optimal resource allocation**	↑	↑	↑

After critically analyzing the past works it has been concluded that many rigorous works havebeen done and practical frameworks and methodologies have been proposed in order to reduce the total incurring latencies and make shorter response time. Many authors have proposed successful approaches in order to enhance the real-time performance of IoT environment. However, it has been found that the decision to offload the task in verycomplex in itself and needs a lot of work in future for an optimal decision model so that the task components are not lost while responding back if partial offloading policy is adopted [39]. In addition, if full offloading is carried out then parameters like ultra-latent latency, balanced workloads and real-time computation should be of critical focus.

3. Proposed Work

In this research study a concrete model for IoT-MEC task offloading is being proposed to resolve the latency issues. The proposed model is based on three constituent technologies namely Internet of Things, Mobile Edge Computing and Cloud Computing. The modelexplains the task offloading methodology through splitting a task into subunits and allocating them to both device and remote area for parallel computation. The selection of remote region will depend upon the task requirements and server status of both MEC servers and Cloud servers. Remote Computation will be carried since the IoT devices are resource constrained devices in terms of storage and computational capabilities. Thus, there is an urge of offloading the task to either the edge servers or local clouds for the purpose of task computation.

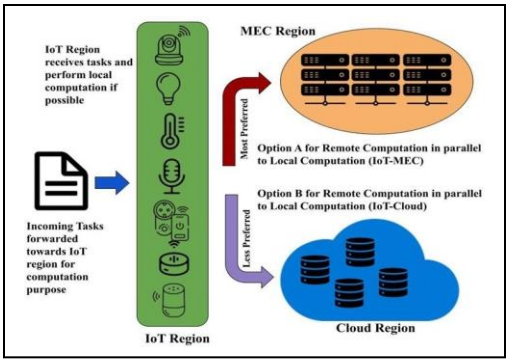
IoT Region receives tasks and perform local computation if possible
Incoming Tasks forwarded towards IoT region for computation purpose
IoT Region
MEC Region
Most Preferred
Option A for Remote Computation in parallel to Local Computation (IoT-MEC)
Option B for Remote Computation in parallel to Local Computation (IoT-Cloud)
Less Preferred
Cloud Region

Figure 2. Figure illustrating an outline of propoed task methodology

The aim is to provide optimal set of resources to these resource-intensive tasks and to reduce latency. If cloud is available locally then the task will be offloaded directly to the cloud, sincecloud is a resource rich region it will always be preferred for task computation. However, if cloud is at long distance then offloading at edge is preferred over it for offloading tasks. Finally if the task computation requirement matches to local device then no offloading will becarried out and task will get locally computed.

A. Introduction and Modeling of Task

There are a number of heterogeneous tasks generated from IoT devices. It may be uploading of documents, audio/video streaming, sensor data processing and others. But basically a Task,may it be of any type, is a single unit of work. This task may either be a sub-part of some complex task or an input for another task computation. In general, any generated task may be categorized as a vector of five possible attributes namely Task ID(Λ_{ID}), Task Size(d_{Λ}),Computational Intensity i.e. number of CPU Cycles in bits(c_{Λ}), Task Deadline(Λ_{TD}) and Task Workload(WL_{Λ}). It may be modeled asΛ(Λ_{ID}, d_{Λ}, c_{Λ}, Λ_{TD}, WL_{Λ}). The taxonomy of the task may be illustrated as Fig 3.The task dependency (Λ_{dep})plays a very important role in task offloading process. A low-dependent task is more highly to get offloaded easily as compared to a coupled and dependent task. The dependency may be due to system internal architecture or the task may be acting as input source to another task. In this case it is highly preferred to run the task locally without offloading since it may create complexities with results. Λ_{dep} Is set to 0 is task is independent and 1 if it is dependent. The factor will be checked before final offloading decision. In other cases, the task may simply be offloaded to either edge server or cloud for parallel computation along with local processing of some part of task. The value of Λ_{dep} will be system dependent. In this research paper only independent task subunits has been considered.

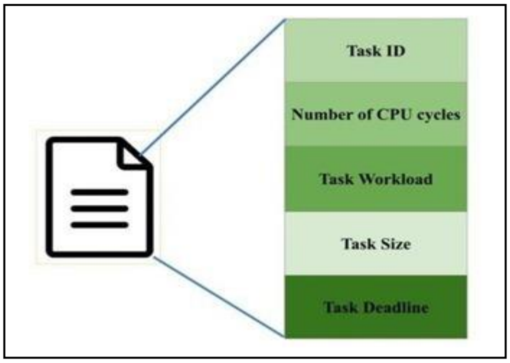

Figure 3. Figure illustrating the Taxonomy of Incoming Task

The total time a task Λ may take to get itself finally computed include the following timelines:

1. Local Time for servicing a Device (T^{i}_{cmp}): This is the total time a task may take to get computed locally.
2. Remote Time for servicing a Device (T^{j}_{cmp}): This is the total time a task may take to get computed remotely.
3. Cloud Time for servicing a Device (T^{c}_{cmp}): This is the total time a task may take to get computed at cloud.
4. Time to transmit a task $(tTrans^{t}_{\Lambda})$: This is the time the task will take to get offloaded to either edge server or cloud.

There should be a time when the decision should be taken to offload the task for remote computation, when local computation only is not enough and deadline of task completion is not met. Thus, it is the responsibility of the device to take the decision within the time before servicing of task is required and task deadline approaches. It is also necessary so that task splitting is done before time and a task unit for both local and remote computation is assignedto both the resources.

The time when the task offloading decision and task spilt decision will be taken by the device will be an internal decision and it may be formulated as:

$$\begin{cases} t_{off} = \Lambda_{TD} - t \\ t_{split} = \Lambda_{TD} - \left(t_{off}/2\right) \end{cases} \quad (1)$$

After the task is split into two subunits, the other sub-unit is required to be allocated to the associated MEC server. Therefore it needs to be transmitted towards the remote server which makes use of available allocated bandwidth Bw_i.Thus, the total time to transmit a task for remote computation may be calculated as:

$$tTrans_{\Lambda}^{t} = \frac{d_{\Lambda}^{remote}}{Bw_i log_2(1 + SINR_i)} \quad (2)$$

where $SINR_i$ is signal to interference plus noise ratio and can be calculated as :

$$SINR_i = \alpha_{ij} \frac{Trans_{ij}Chg_{ij}}{\sum_{i \in U, j \in N} Trans_{ij}Chg_{ij} + \partial} \quad (3)$$

where $Trans_{i,j}$ is the transmit power of device i , $Chg_{i,j}$ is the channel gain between device i and server j, and ∂ is the interfering noise factor. It is necessary to calculate the noise since there may be hundreds of Iot devices which is connected to a single MEC server.

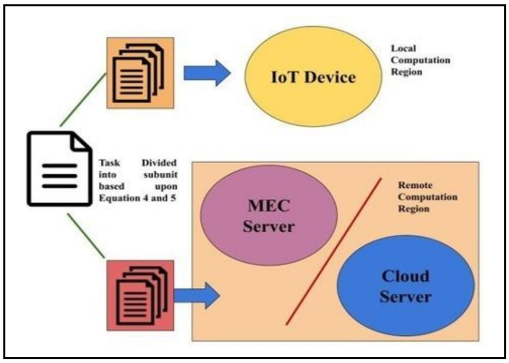
IoT Device
Local Computation Region
Task Divided into subunit based upon Equation 4 and 5
MEC Server
Remote Computation Region
Cloud Server

Figure 4. Figure Illustrating the Task Splitting Procedure

Therefore, the transmission of data from one device will be getting influenced by the transmission of another task for computation at the same resource. In further sub-sections it has been discussed about the criteria of task offloading which will be totally based upon parametric comparisons and evaluations with splitting task for parallel remote and local computation.

Since the total task will be split into two subunits, it will affect the total task size allocated forcomputation at local and remote regions with partial task offloading. It is assumed that Task splitting is an internal decision to be taken at device level thus to check the pivot point a random value P is taken which will decide the limit to which task subunit it is possible for local processing. The total size the task will be divided into 1 subunit each of any memory unit considered. For example if the task size is of 270 GB then it will be considered as a composition of 27 units each of 10GB and vice versa. This may be described as:

$$Pn_{packets} = {}^{d_{\Lambda}}/_{10} \quad (4)$$

The value of P will depend upon the task, device and server status and will be purely hardware and network based. In this paper a formula to set the pivot point is described as:

$$P = {}^{n_{packets}}/_{\Lambda_{TD}} * 2 \quad (5)$$

Until P the task subunits will be allocated for local computation and the remaining unit after P will be offloaded for remote computation. The data size will also break here thus into two parts which may be formulated as:

$$\begin{cases} d_{\Lambda}^{local} = P \\ d_{\Lambda}^{remote} = n_{packets} - d_{\Lambda}^{local} \end{cases} \quad (6)$$

The d_{Λ}^{local} data size of the divided task will be allocated for local computation while d_{Λ}^{remote} data size will be allocated for remote execution at the associated remote server. The total remote and local computation times may be calculated as follows:

$$T_{cmp}^{j} = tTrans_{\Lambda}^{t} + \frac{(d_{\Lambda}^{remote})}{T_{wait}^{j}} \quad (7)$$

The goal is to minimize the total computation time and reduce the total response time for the computed task in a real-time manner. Similarly if the task gets computed locally without any offloading then it may be computed as:

$$T_{cmp}^{i} = \frac{(d_{\Lambda}^{local})}{T_{wait}^{i}} \quad (8)$$

If the task splitting process is carried out then the total computation time can be calculated as:

$$Tcmp = \left(T_{cmp}^{j} - T_{cmp}^{i}\right) \quad (9)$$

In above equation a subtraction operation has been performed between two timelines since computation of both parts of the task will be in parallel and not procedural.

Thus the remote time will be total time excluding the local computation time.
Lastly the total workload imposed by the task may be calculated as follows:

$$\mathrm{WL}_{\Lambda} = d_{\Lambda} * \left(\frac{c_{\Lambda}}{\Lambda_{TD}}\right) \tag{10}$$

The goal is to minimize the total computation time for the generated task as well as to send back the computed results with optimal computation time.

B. Device Modeling

The IoT device is the point from where random and heterogeneous tasks will be generated for computation. Suppose there are U IoT-nodes and each device may be represented as i^{th} *IoT device* where $i \in U$. Any i may be categorized as a vector of five possible attributes namely Device ID (i_{ID}), Device's computational capability (Cap_{i}), Total allocated Bandwidth (Bw_{i}), Energy Consumption (E_i) and Workload (WL_i). It may be modeled as $\mathrm{i}(\mathrm{i}_{ID}, Cap_{\mathrm{i}}, Bw_{\mathrm{i}}, \mathrm{E}_i, \mathrm{WL}_i)$. Through i_{ID} each device may be identified uniquely in the IoT-MEC environment and will be used to know the device status. Cap_{i} of device i is a result of four attributes which assists to get the knowledge of current device capacity that is the Processing time (busy + idle), memory status, queue length avg. and network consumption. The total allocated bandwidth to any i^{th} device is calculated as a product of number of tasks generated by the device and throughput of the device.

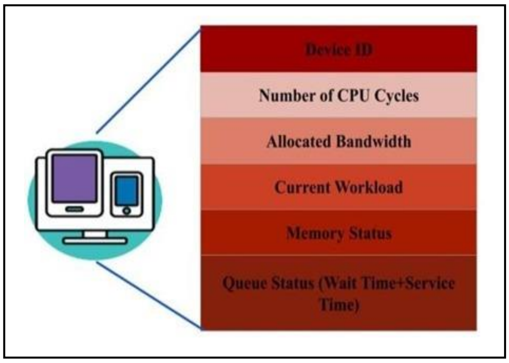

Figure 5. Figure Illustrating Taxonomy of IoT Devices for Task Offloading

Calculation of energy consumption is a critical factor to consider verifying the efficiency and data migration within the network, since these IoT devices are power constrained in nature with limited battery life. In further an analysis of energy consumption will be done for all the three involved tiers. And, finally the workload over the device may be defined as the amount of tasks to be done by the computing resources of the device, allocated over it. The workload of the device WL_i may be represented as:

$$\mathrm{WL}_i = \sum \mathrm{WL}_{\Lambda} \tag{11}$$

Each i^{th} device will have some threshold capacity in order to process the offloaded tasks and to compute the offloaded workloads. Let it be denoted by WL^{i}_{max}which the maximum is allowed workload. This is the maximum amount which a device may process locally. The remaining space for arriving load to get processed may be calculated as:

$$WL^{local}_{rem} = WL^{i}_{max} - \mathrm{WL}_i \tag{12}$$

If any task with workload exceeding WL_{rem} should be split and offloaded towards the server for computation.

C. Server Modeling

This section includes the modeling of both the edge server and cloud server. In general, the server model may be presented as $\mathrm{j}(\mathrm{j}_{ID}, Cap_{\mathrm{j}}, Bw_{\mathrm{j}}, \mathrm{E}_j, \mathrm{WL}_j)$. The tuple-based presentation is not mentioned for Cloud since the region has ample resources for storage and computation. It is understood that the task will surely get computed within some instants of time if offloaded to cloud.

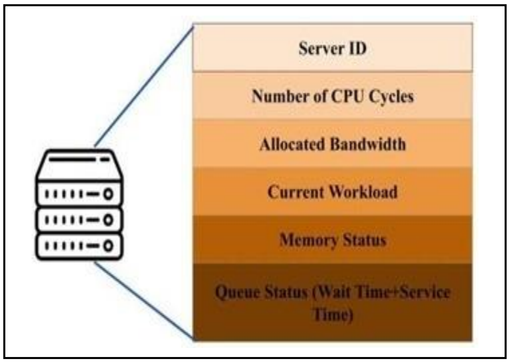

Figure 6. Figure Illustrating Taxonomy of Edge Server for Task Offloading

The proposed model will include two groups of classes. One will be N MEC servers where each server $j \in N$ and U IoT-nodes of User-end mobile devices where $i \in U$. The participants of the former class will be fixed at some position but the latter participants will be mobile in nature which means they will not stay in a fixed or central location but will be moving continuously. Due to this scenario, any i^{th} device will be near to a given j^{th} server at a given instant time t, but may get far from the discovery of j^{th} server at time $t + t'$. All the N MEC servers are geographically distributed within the MEC region. They are interconnected with each other, along with its connection to hundreds of U IoT-nodes and for obvious reason it will be also connected to central Cloud Server *C*. The connected IoT nodes will be of heterogeneous nature and will have their own processing and QoS requirements.

Also the communication frequencies of a $i - j$ pair will differentiate when compared to $j - j$ pair. The workload of the device WL_i may be represented as:

$$\mathrm{WL}_j = \sum \mathrm{WL}_\mathrm{i} \tag{13}$$

Similar to the device, each j^{th} server will also have some threshold capacity in order to process the offloaded tasks and to compute the offloaded workloads. Let it be denoted by WL_{max}^{j}which the maximum is allowed workload. This is the maximum amount which a server may compute remotely. This is the maximum amount which a server may process remotely. The remaining space for arriving load to get processed may be calculated as:

$$WL_{rem}^{remote} = WL_{max}^{j} - \mathrm{WL}_i \tag{14}$$

If any task with workload exceeding WL_{rem} should be split and offloaded towards the server for computation.

For the sake of task offloading, it is necessary that the device should be connected to a dedicated server, so that the data may be routed to exact destination for computation. In order to verify the connection between the two classes, an association indicator will be assumed as $\alpha_{ij}\ where\ \alpha_{ij} \in [0,1]$.If $\alpha_{ij} = 1$ will then there exists a successful connection otherwise $\alpha_{ij} = 0$ will tell the non-association of the two parties. Another association indication will also be considered to connect device directly with cloud as $\alpha_{ij}\ where\ \alpha_{iC} \in [0,1]$.If $\alpha_{iC} = 1$ will then there exists a successful connection otherwise $\alpha_{ij} = 0$ will tell the non-association of the two parties. The distance based partial task offloading has been discussed in further sections of this paper.

4. POLICIES FOR TASK OFFLOADING

A. Distance-Based Task Offloading

The goal is to find and associate a device *i* with server *j* that could possible minimize the execution and transmission latency with optimal resource utilization. In order to do the mentioned the following distance formula is used which will take the x and y co-ordinates of the geographically located device and server in order to calculate the distance as following:

$$t_{dist} = \sqrt{(i_{x2} - j_{x1})^2 + \left(i_{y2} - j_{y1}\right)^2}, t_{dist} < t_{dist}^{max} \tag{15}$$

where t_{dist}^{max} is the maximum discoverable allowed distance between $i - j$ pair.

The following constraints have been set to accomplish the goal of minimum uplink and downlink transmission latency. The constraints will help to verify if the device has been associated with the server which exist at nearest distance, so that if offloading of task occurs, then it take less time to migrate the task for computation as well as to deliver computed result. Following are the specified constraints:

$$Cons\ A\colon min \sum_{u \epsilon U,} \sum_{j \epsilon N} (t_{dist}) \tag{16}$$

$$Cons\ B = \sum_{j \epsilon N} \alpha_{ij} \epsilon \{0,1\} \tag{17}$$

The distance should be less than the maximum threshold distance t_{dist}^{max} to avoid long-term latency and bad response time for real-time requests. The association request is first sent to the MEC orchestrator which decides either to allocate the selected resultant MEC server or not.

B. Task Requirement-Based Offloading

This section discusses about the task execution based on task offloading process. The task will be executed on the basis of some task requirement criteria which is needed to get fulfilled before execution starts. It is not necessary that a task can fully be computed at one place. It is possible to divide the task and compute it either locally or remotely at edge server or cloud or both. This depends upon the resource requirements of the task. IF task requirements are very low then it is possible to compute it locally. However, if the requirements are too high, then it should be offloaded after splitting towards either edge server or cloud. However, the execution time becomes variably high whenever the task gets offloaded to the cloud.

The amount of memory currently being allocated may be calculating by the difference of total provided memory and total memory in use. Let both be represented as M_{free} and M_{alloc} thus total memory M_{total} may be given as $\mathrm{M}_{total} = \mathrm{M}_{free} + \mathrm{M}_{alloc}$. Concurrently, M_{free}^{l} and M_{free}^{r} represent free memory for device and server processor. Each unit of task Λ will be assigned for local or remote processing. The execution preference is set to local computation but, will move to edge server computation mode based upon the requirement of task. Let Λ=[$\Lambda_1, \Lambda_2, \Lambda_3, \ldots\ldots, \Lambda_n$] be the total number of sub-units that will computed locally and remotely. The data units of total task to be processed, exists in form of continuous and systematic data pieces. Each sub unit is a continuous and formal piece of data which is independent to either get processed locally or at server or finally at cloud.

C. Server Status-Based Offloading

This section discusses about the task splitting procedure on basis of server status. The total number of task units in a randomly generated task will be divided with some typical method for parallel remote and local execution. The goal to carry out this process is to minimize the total execution latency and energy consumption with optimal resource allocation. For this purpose three sets have been designed namely $\mathrm{L}_x = \{\wedge_1^l, \wedge_2^l, \ldots\ldots, \wedge_n^l\}$, $\mathrm{R}_y = \{\wedge_1^R, \wedge_2^R, \ldots\ldots, \wedge_n^R\}$, $\mathrm{DC}_z = \{\wedge_1^C, \wedge_2^C, \ldots\ldots, \wedge_n^C\}$ and containing those subunits of task $\wedge$ assigned for local, remote and cloud based computation.

The size of total task may be represented as $\mathrm{L}_x + \mathrm{R}_x$ in case of server association while $\mathrm{L}_x + \mathrm{DC}_x$ in case of Cloud association. It may e formulated as:

$$d_\wedge = (\mathrm{L}_x + \mathrm{R}_x + \mathrm{DC}_x).(\,\alpha_{iC}.\alpha_{ij}) \qquad (18)$$

However, the final decision to allocate the resources for task execution depends upon the server status. The goal of this paper is to minimize the total energy consumption and to reduce the total computation time and transmission time. The first goal that is minimum transmission latency, the second goal that is to minimize the total execution latency and finally the third goal to provide optimal resource utilization. In the next section the proposed model will be validated on the basis of hypothesis based statistical analysis tests in order to verify its performance and accuracy to validate the proposed objectives.

5. ULTRA-LATENT TASK OFFLOADING IOT-MEC MODEL:VALIDATION APPROACH

In this section the proposed model will be validated on the basis of hypothesis based statistical analysis tests in order to verify its performance and accuracy to validate theproposed objectives.

A. Data Collection

In this paper the data for the validation of the proposed model has been collected from two different datasets [41, 42] which best fits the proposed idea and the mentioned objectives of research. Both the dataset have been combined for this research paper and the validation of the proposed method and maximum attributes from both the datasets have been selected as per the requirement for validation and verification of algorithm. The following Table 3 and Table 4 describe details of dataset attributes with description:

TABLE III.

ATASET TYPE 1

ID	CurrentAllocatedSize
Size	CurrentAllocatedRam
Name	CurrentAllocatedBw
MIPS	CurrentAllocatedMips
NumberOfPes	BeingInstantiated
RAM	GeoLocation/Latitude
BW	GeoLocation/Longitude
Source	DataType
Destination	DataPercentage
Delay	Tuple_Reversed
Priority	IsServerFound
CloudletScheduler/	IsCloudServed
PreviousTime	IsServed
ClouletScheduler/CurrentMips	DeviceType
Service	IsServedByFC_Cloud
QueueDelay	BurstTime
InternalProcessingTime	BurstTimeDifference
FogLevelServed	IsServedByFC (output)

For workload allocation on CPU when task has been finally offloaded to the remote regions either MEC server or Cloud server, the dataset from latter source will be taken into consideration to verify optimal CPU utilization, network utilization and workload computation low latency and quick response and execution time. The following Table 4 represents the attributes of dataset taken for the purpose.

TABLE IV. D

ATASET TYPE 2

Job Number	Requested Memory
Submit Time	Status
Wait Time	User ID
Run Time	Group ID
No. Of Allocated Processors	Executable (Application) Number
Avg. CPU Time Used	Queue Number

Used Memory	Partition Number
Requested Number of Processors	Preceding Job Number
Requested Time	Think Time for Preceding Job

The tasks as mentioned in previous sections, will arrive at IoT device initially and from here a decision will be taken whether the task will be computed locally or should be allocated to a remote server. These devices and servers are geographically distributed and the tasks is in form of tuples. The geo-locations are of random nature and may be present at random places. The dataset from source 2 contains a large amount of worklog files. These log files contains data from a number of datacenters distributed geographically across the globe. Some datacenters included in these log files are NASA iPSC, KTH SP2, Sandia Ross, SDSC DataStar and many more. The logfiles also serve as a raw data volume in order to design a number of workload models. The log files contains data taken for a specific amount of duration from mentioned datacenters with raw and processed data collection in a format as Number of Jobs, Number of Users and % utilization of CPU.

In order to execute the algorithm and analyze its accuracy, performance and efficiency, it is required that only important attributes to be considered leaving behind those parameters that are not useful. These additional parameter may also affect the final result negatively or influence the expected output, therefore the procedure of detecting and removing outliers must be done beforehand. The dataset must contain all the data in numeric form but it has been observed that the dataset from source 1 contains the following attributed in categorical form:

1. Priority: High, Low, Medium
2. Data Type: Abrupt, location-based, multimedia, medical, textual.
3. Device Type: Actuator, dumb Object, mobile, sensor, node

Finally the following table present all the selected attributes from both the datasets combined into one to validate the proposed method. The table differentiates the selected attributes in the form of dependent and independent variables so that further various statistical tests may easily be applied over comparative and isolated parameters.

TABLE V.
COMBINED DATASET FOR DEPENDENT VARIABLES

Dependent Variables	
RAM	Service
BW	QueueDelay
CurrentAllocatedSize	InternalProcessing Time
CurrentAllocated	FogLevelServed
Ram	
CurrentAllocated Bw	IsServedByFC_Cloud
CurrentAllocated Mips	BurstTime
Tuple_Reversed	BurstTime Difference
IsServerFound	IsServedByFC (output)
IsCloudServed	Wait Time
IsServed	Run Time
Requested Memory	No. Of Allocated Processors
Requested Time	Avg. CPU Time Used
Used Memory	Requested Number of Processors

TABLE VI.
COMBINED DATASET FOR INDEPENDENT VARIABLES

Independent Variables
ID
Size
CloudletScheduler/PreviousTime
BeingInstantiated
Queue Number
GeoLocation/Latitude
GeoLocation/Longitude
ClouletScheduler/ CurrentMips
Submit Time
DataPercentage

The selected attributed will be imposed for the collected data over it for the proposed model in order to check its accuracy, performance and effectiveness over minimizing total incurred latency during task execution and task offloading with energy consumption. In the upcoming subsection various tests have been applied to the dataset on the basis null and alternate hypothesis through which the successful validation of the proposed solution may be represented to accomplish the goals of latency and energy minimization with optimal resource allocation.

B. Validation Tests

A number of hypothesis tests have been applied which successfully validates the proposed algorithm and solution in terms of reducing the total computational latency of the task. The hypothesis have undertaken four policies to represent the significant reduction in task computation, namely

- ***Policy 1:*** Full Offloading to Cloud

- ***Policy2:*** Full Offloading to MEC server
- ***Policy 3:*** Local computation without offloading
- ***Policy 4:***Partial Offloading to with both local and remote computation of task(Proposed Policy).

The following section consists various hypothesis tests through which the proposed algorithmwill be tested for its effect on minimizing the total computation latency. In order to verify the proposed algorithm, two hypothetical statements are formulated which will be tested and verified against each mentioned tests as follows:

***H*₀:** Full offloading of task for remote execution at the edge server is the only solution to deliver fast responses and meet real-time scenario in IoT environment.

H1: Partial task offloading with task size splitting will reduce the total task computation time will quicker request responses and efficient performance of IoT environment.

1) Shapiro Wilk Test

This test is used to analyze the normality of the data distribution related to a continuous variable. It states two statements in the form of null and alternate hypothesis which says:

H_0: The variable is abnormally distributed.

H_1: The variable is normally distributed.

If the p-value is less than 0.05 then null hypothesis is rejected otherwise accepted. For the proposed method, the data of total time of task computation over all the four policies will be analyzed for its normality using the test. The following table will help to validate thenormality of data for all the four policies and their related computation time.

TABLE VII.
ANALYSIS OF TASK OFFLOADING POLICIES FOR NORALIY CHECK

	Policy 1	**Policy 2**	**Policy 3**	**Policy 4**
P-Value	0.1437	0.1437	0.2625	0.6949
W	0.883	0.883	0.9063	0.9483
Sample Size	10	10	10	10
Average	62670.7494	41780.7996	13914.4828	5485.62
Median	51720.0841	34480.0561	11166.6667	5243.5578
Standard Deviation	39499.002	26332.668	6896.9227	1980.0868
Skewness	0.9832	0.9832	0.8487	0.7691
H_o Accepted?	Yes	Yes	Yes	Yes
Excess Kurtosis	0.01472	0.01472	-0.1896	0.667

Since a larger P-value supports the null hypothesis, it may be analyzed that all the four policies is larger than 0.05 which supports the null hypothesis. From here, it may be concluded that the data is normally distributed.

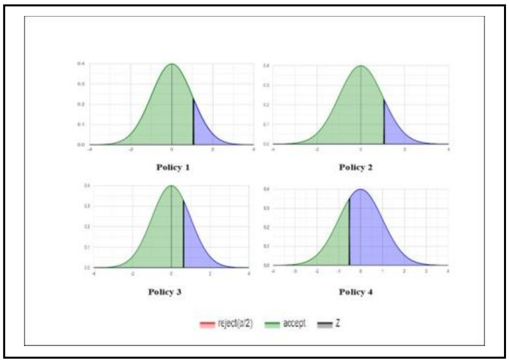

Figure 7. Figure Illustrating Normality of Data Distribution for all four Policies

Finally, it has become clear that all the data is normally distributed and there are no abnormalities present which could be removed or corrected. Through the test it has been successfully validated that the samples from the dataset relating to task computation time of different regions like local, MEC, Cloud and Local with MEC, are normally distributed sample sets.

2) Two-Sample Z-Test

It is a statistical hypothesis testing technique which is used to verify the equality between twogiven population sets. The test can be carried out nly if standard deviation of both the population are known beforehand. Moreover, to have this test result in success, the first factorto check is the normal distribution of data. Although in the previous section, it has already been verified through Shapiro Wilk Test that the considered dataset is normally distributed.The following formula is used to carry out this test:

$$\frac{(\overline{x_1}-\overline{x_2})-(\mu_1-\mu_2)}{\sqrt{\frac{\sigma_1^2}{n_1^2}+\frac{\sigma_2^2}{n_2^2}}} \tag{19}$$

where, $\bar{x}_1$ and $\bar{x}_2$ are respective means of both samples, $\bar{\mu}_1$ and $\bar{\mu}_2$ are respective means of both populations, σ^2 and σ^2 are standard deviations and n_1 and n_2 are respective data points. For the proposed algorithm, a sample size of 10 will be taken and through this the test will be carried out. The following results show the successful rejection of null hypothesis for significance level 0.05.

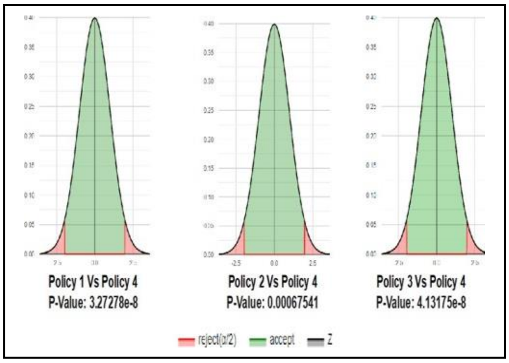

Figure 8. Figure Illustrating Normality of Data Distribution for all four Policies

Thus, the test accepts the alternate hypothesis statement that it will be very much feasible to split the incoming tasks from user-end devices into local and remote computation areas and carry out task execution parallel at both sides, which will result in reduced latency in total task computation time.

3) F-Test

F-test is a way to test given hypothesis on the basis of variances of given sample populations. The equality of variances is tested followed with f-distribution and uses f-statistic in order to test its equality. In order to carry out an f-test, the sample population must be a set of independent events. After conducting the f-test the results are analyzed if it is statistically significant. If yes, then the null hypothesis is rejected otherwise it is accepted. In this paper, two-tailed F-Test has been considered where,

$$\mathbf{H_0}: \sigma_1^2 = \sigma_2^2$$

$$\mathbf{H_1}: \sigma_1^2 \neq \sigma_2^2$$

Decision Criteria: Test-Statistic F > F-Test Critical Value

The test-statistic F is calculated as F= σ_1^2/σ_2^2. The F-Critical value is a data point which is used to decide the acceptance or rejection of null hypothesis.

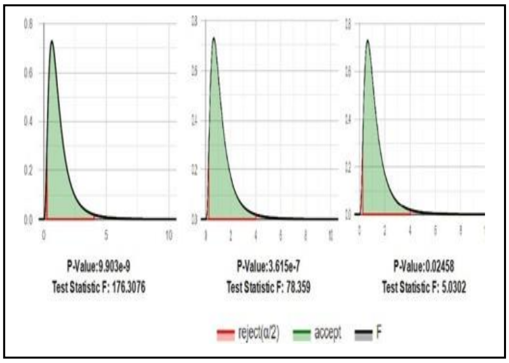

Figure 9. Figure Illustrating F-Test Results

For the considered dataset of this paper the acceptance region falls between [0.2484:4.026].In the figure above it has been analyzed that the p-value never falls between the acceptance range and thus null hypothesis will be rejected. This again verifies the validity and performance of the proposed algorithm.

4) Multiple Linear Regression Test

This is a statistical technique which may be used for the prediction of some output variable based upon the population samples of two or more variables. The outcome or variable to be predicted is known as dependent variable while the variables over which this prediction iscarried out is known as independent or explanatory variables. The formula for calculating multiple linear regression is as follows:

$$Y = b_0 + b_1X_1 + b_2X_2 + \cdots + b_pX_p + \in \tag{20}$$

where, Y is the outcome or predicted variable, b_0 is the y-intercept, b_1 and b_2 are regression coefficients, b_p is the slope coefficient for each independent variable and $\in$ is the model's (residual) error factor. The Test statistic F may be calculated as F=$(Reg)/MS(Res)$, where Reg means regression and Res means Residual. The null and alternate hypothesis may be given as follows:

$$\mathbf{H_0}: b_0$$

$$\mathbf{H_1}: b_0 + b_1X_1 + b_2X_2 + \cdots + b_pX_p + \in \tag{21}$$

Decision Criteria: p-value < significance value (to reject

H_0)

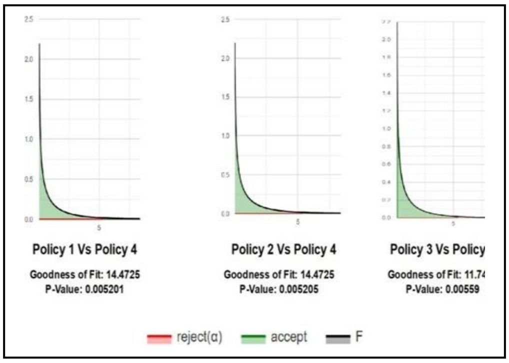

Figure 9. Figure Illustrating Multiple Regression Test Results

In the above figure it has been analysed that in all the three policies as compared against the proposed policy, the p-value is found to be less than the significance level 0.05. Therefore, this test also verifies the validity and performance of the proposed task offloading policy.

5) One-Way ANOVA and Tukey HSD Test

This is another very effective statistical method or technique to test a given hypothesis, whichverifies the hypothesis proposed on the basis of difference between the averages of given sample populations and test its significance. In one-way ANOVA test, the F-statistic value is calculated as the ratio of variance between as well as inside the sample populations. The smaller is the value of F, the more likely are their equal averages. The null and alternate hypothesis may be given as:

$$\textbf{Ho: } \mu_1 = \cdots = \mu_k \qquad \textbf{H1: } no(\mu_1 = \cdots = \mu_k) \tag{22}$$

Decision Criteria: P-value>Test-Statistic F

The Test statistic F may be calculated as F=MSG/MSR, where Reg means regression and Res means Residual. The degree of freedom may be calculated as $k-1$ and related error as $n-k$. The sum of squares may be calculated as follows:

$$SSG = \sum_{i=1}^{k} n(\bar{x}_i - \bar{x})^2 \tag{22}$$

$$SSE = \sum_{i=1}^{k} (n_i - 1)^2 \tag{23}$$

Thus, the MSG and MSR may be calculated on the basis of eq 24 as follows:

$$MSG = SSG/(k-1) \tag{24}$$

$$MSE = SSE/(n-k) \tag{25}$$

where, k is the total number of groups or sample considered, n_i is the sample side of group i, n is the overall sample side, $\bar{x}_i$ is the average of group i, $\bar{x}$is the overall average and S_i is the standard deviation.

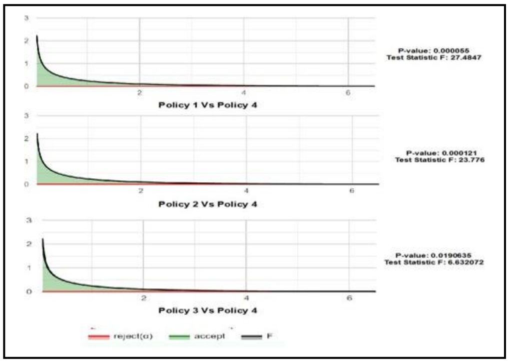

Figure 10. Figure Illustrating One-Way AMOVA Test Results

For the considered dataset of this paper the acceptance region falls between [∞:4.4139]. In the figure above it has been analyzed that the p-value never falls between the acceptance range and thus null hypothesis will be rejected. Also, In the above figure it has been analysed that in all the three policies as compared against the proposed policy, the p-value is found to be less than the significance level 0.05. Therefore, this test also verifies the validity and performance of the proposed task offloading policy.

The following algorithm describes the proposed model intended to perform the mentioned objectives of the research paper.

```
Algorithm: NTRMLEC Algorithm
INPUT: Set of IoT Nodes $i \in U$, Set of MEC Servers $j \in N$.
        $i = \{i_1, i_2, i_3, \ldots\ldots, i_n,\}$ and $j = \{j_1, j_2, j_3, \ldots\ldots, j_n,\}$
        Set of tasks $\Lambda_i$=[$\Lambda_1, \Lambda_2, \Lambda_3, \ldots\ldots, \Lambda_n$] $n \in \Lambda$
        Set of local processing subunit $L_x = \{\Lambda_1^l, \Lambda_2^l, \ldots\ldots, \Lambda_n^l\}$,
        Set of remote processing subunits $R_y = \{\Lambda_1^R, \Lambda_2^R, \ldots\ldots, \Lambda_n^R\}$ and
        Set of Cloud processing subunits $DC_z = \{\Lambda_1^C, \Lambda_2^C, \ldots\ldots, \Lambda_n^C\}$
OUTPUT: Nearest Distance Task-Resource
        Allocation with minimized latency
```

```
and Energy Consumption
1.START
2.Calculate number of packets of task $\Lambda_i$ using eq 4
3.Let TS(t)= $\Lambda_i$(t)
4.WHILE TS(t) ≤ $n_{packets}$
5. DO
      Take Prior knowledge of $Cap_{\mathrm{i}}, Bw_{\mathrm{i}} Cap_{\mathrm{j}}, Bw_{\mathrm{j}}$ of device and Server
      Calculate the workload $\mathrm{WL}_i$, $\mathrm{WL}_j$ of device and server by eq 12 and 14.
      Calculate the remaining processing capacity of both device and server using eq 13 and 15.
      Check the task status $\Lambda(\Lambda_{ID}, d_\Lambda, c_\Lambda, \Lambda_{TD}, \mathrm{WL}_\Lambda)$ , device status $\mathrm{i}(\mathrm{i}_{ID}, Cap_{\mathrm{i}}, Bw_{\mathrm{i}}, \mathrm{E}_i, \mathrm{WL}_{\mathrm{i}})$ , server status $\mathrm{j}(\mathrm{j}_{ID}, Cap_{\mathrm{j}}, Bw_{\mathrm{j}}, \mathrm{E}_j, \mathrm{WL}_j)$ using mentioned euqations
      Perform the following parametric comparisons for each task subunit of task $\Lambda_i$.
           i.   FOR k=0 to n-1
           ii.  Calculate Total Transmission Time using Eq2
           iii. IF $T^i_{wait} < \Lambda_{TD}$ AND $\mathrm{WL}_i < WL^{local}_{rem}$ AND $d_\Lambda < M^l_{free}$
                     Allocate task subunits to $\mathrm{L}_x$
                     Calculate Total Local Computation Time using Eq8
                     GOTO HYP

                ELSE IF $T^j_{wait} < \Lambda_{TD}$ $\mathrm{WL}_j < WL^{remote}_{rem}$ AND $d_\Lambda < M^r_{free}$
                     Discover j* on basis of eq 21.// for maximum 5 servers
                     For ith make a distance-based set of j* as $t^j_{dist} = \{d_1, d_2, d_3, \dots\dots, d_n\}$
                     Verify $min(t^j_{dist}) < t^{max}_{dist}$
                     Otherwise GOTO Step A
                     Assign j with $min(t^j_{dist})$ to ith device.
                     Verify $Cons\ A\ and\ Cons\ B$ by eq 22 and 23
                     Otherwise GOTO Step A
                     Set $\alpha_{ij} = 1$
                     Calculate Pivot Point and Data sizes of splitted data using Eq 5 and 6
                     Split task subunits to $\mathrm{L}_x$ until $P$ and remaining to $\mathrm{R}_y$
                     Calculate $Tcmp$ using eq 7 and 9
                          $results = \mathrm{L}_x \cup \mathrm{R}_y$
                     GOTO HYP

           i.   ELSE
                     Set $\alpha_{ic} = 1$
                     Calculate Pivot Point and Data sizes of splitted data using Eq 5 and 6
                     Split task subunits to $\mathrm{L}_x$ until $P$ and remaining to $\mathrm{DC}_z$
                     Calculate total computation time.
                          $results = \mathrm{L}_x \cup \mathrm{DC}_z$
                     GOTO HYP
                ENDIF
           ENDFOR
HYP: Perform-
                     1. Shapiro-Wilk Test
                     2. Two-Sample Z-Test
                     3. F-Test
                     4. Multiple Regression Test
                     5. ANOVA and Tukey HSD
6.STOP
```

6. RESULTS ANALYSIS

After the successful hypothesis testing on the proposed task offloading policy, this section aims to visualize the final findings and result interpretation over the effect of proposed methodology with respect to reduced latency in IoT-MEC environment. In Fig 9 it may be analyzed that the proposed task offloading policy is the best fit as compared to other existing task offloading policy. It may be seen that the first task offloading method that is the cloud- based task offloading has the highest latency in terms of response time.

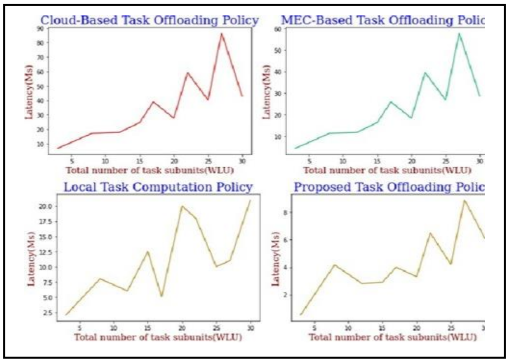

Figure 11. Task Computational Latency for adopted policies

The reason is that whole task size of incoming tasks are sent to these cloud-based serverwhich have high queuing and downloading time which add up to high response time. Nextthe MEC-server based task offloading policy where whole tasks are sent to the nearest edge servers. These also have considerable queuing time which results in not matching to response for real-time scenario. If the task is computed locally with offloading it to server or remotecomputation which is the third policy, it may be possible that these devices do not have muchresources and power which may lead to high response time. However, if the task is divided into subunits and parallel assigned to local as well as remote computation, which is the proposed task offloading policy, then it is highly possible that the request and task computation will be completed within few instants of time with overall reduced latency and quick response time. Next in Fig 13. It has been illustrated the performance of each policy and compared with the proposed policy. It has been that the proposed policy has the least computation time as compared to the other three policies.

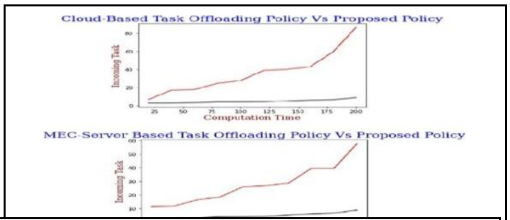

Figure 12. Incoming Task Traffic Vs. Computation Time

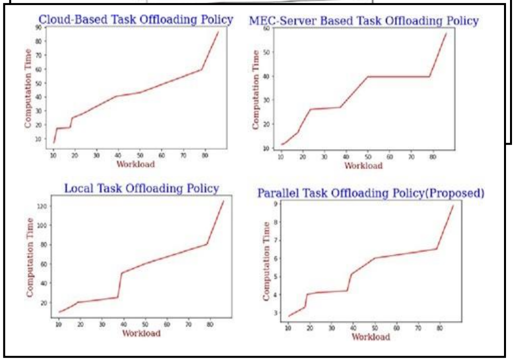

Figure 13. Task Computation Vs Total Workload

Next in Fig 14, it has been illustrated the effectiveness of each existing as well as proposed algorithm for computing the workload within a given time window. It was found the least computation time is taken by the proposed policy which supports both local as well as remotecomputation.

Finally, from all these results interpretation, it has been found that the proposed policy is capable of reducing the overall computation latency and supports quick response time which is a stringent requirement for IoT real-time scenario. The major role of the proposed policy is to highly support the task splitting and task offloading decision which is the main factor, to be focused when dividing the whole task into subunits for execution. The task consists of many dependencies which should be kept in mind before final division decision and resource assignment. However, the proposed algorithm do focus upon taking an appropriate task offloading decision before final task splitting take place. The proposed algorithm and task offloading method was found to be highly reducing the data traffic over edge servers and minimal computation time and maximum utilization of resources.

7. DISCUSSION

The proposed research has focused upon minimizing latency during transmission of data. Theproposed task offloading method was found to be effective in minimizing various kinds of inter-dependent latencies like transmission latency, computation latency, and task execution latency. This research paper gives a threefold solution along with optimal task offloading and resource allocation decisions based upon device, server and task status. The research has been validated using various kinds of hypothesis tests and was successful in nullifying the null hypothesis for each test. The results have been interpreted and analysed using a powerful programming language Python which is a best for analysing datasets and visualizing the effectiveness of data in various approaches. The two datasets were combined taking a numberof independent and dependent variables and the proposed algorithm was implemented over Anaconda platform. Lastly, the proposed research was successful in resource utilization and minimizing the total incurred latency.

8. CONCLUSION AND FUTURE WORK

Through this research study it may be concluded that there is a heavy need of such kind of mechanism which deals with the mentioned issues and challenges. A qualitative and quantitative defined need of a hybrid approach may be the best fit solution for minimizing latency and maximizing resource utilization and various affecting factors of better computation The proposed method was found to be highly accurate in accomplishing the goalof achieving the affecting factors as compared to the unary methods that were proposed in thepast since they focused upon only one to two parameters. However, the proposed research has focused upon four most important parametersi.e. latency, task offloading and resource allocation, which needs to be upon their optimal status for a maintained IoT-MEC communication internetwork.

In the future it will tried to implement the proposed research in the form of genetic algorithm and will definitely try to implement the data through machine learning approach with real- time parameter and under strict latency and QoS constraints. In future the focus will be upon dependant task whose splitting is a complex process so that its execution may be carried out with minimal latency and energy constraints. This work will be done to effectively enhance the accuracy and effectiveness of the proposed method of latency minimization and resource allocation.

REFERENCES

[1] S Wang, C Fan, CH Hsu, et al., "A Vertical Handoff Method via Self-Selection Decision Tree for Internet of Vehicles," IEEE Systems Journal, vol. 10, no. 3, 2016, pp. 1183-1192.

[2] A Zhou, S Wang, B Cheng, et al., "Cloud Service Reliability Enhancement via Virtual Machine Placement Optimization," IEEE Transactions on Services Computing, 2016.

[3] S Wang, A Zhou, CH Hsu, et al., "Provision of Data-Intensive Services Through Energy- and QoS-Aware Virtual Machine Placement in National Cloud Data Centers", IEEE Transactions on Emerging Topics in Computing, vol. 4, no. 2, 2016, pp. 290-300.

[4] Lin, L.; Liao, X.; Jin, H.; Li, P. Computation Offloading Toward Edge Computing. Proc. IEEE 2019, 107, 1584–1607.

[5] Kuang, L.; Gong, T.; OuYang, S.; Gao, H.; Deng, S. Offloading Decision Methods for Multiple Users with Structured Tasks in Edge Computing for Smart Cities. Future Gener. Comput. Syst. 2020, 105, 717–729.

[6] C. Swain, "METO: Matching-Theory-Based Efficient Task Offloading in IoT-Fog Interconnection Networks." IEEE Internet of Things Journal, vol. 8, no. 16, pp. 12705- 12715, 2021, doi: 10.1109/jiot.2020.3025631.

[7] Ali, N. Riaz, M. I. Ashraf, S. Qaisar, and M. Naeem, "Joint Cloudlet Selection and Latency Minimization in Fog Networks." IEEE Transactions on Industrial Informatics, vol. 14, no. 9, pp. 4055-4063, 2018, doi: 10.1109/tii.2018.2829751.

[8] J. Xue and Y. An, "Joint Task Offloading and Resource Allocation for Multi-Task Multi- Server NOMA-MEC Networks." IEEE Access, vol. 9, pp. 16152-16163, 2021, doi: 10.1109/access.2021.3049883.

[9] Rafiq, W. Ping, W. Min, S. H. Hong, and N. N. Josbert, "Optimizing Energy consumption and Latency based on computation offloading and cell association in MEC enabled Industrial IoT environment." 2021 6th International Conference on Intelligent Computing and Signal Processing (ICSP), 2021, doi: 10.1109/icsp51882.2021.9408693.

[10] S. Xia, X. Wen, Z. Yao, Y. Li, and G. Wang, "Dynamic Task Offloading and Resource Allocation for Heterogeneous MEC-enable IoT." 2020 IEEE/CIC International Conference on Communications in China (ICCC), 2020, doi: 10.1109/iccc49849.2020.9238863.

[11] S. K. T., "EFFICIENT RESOURCE ALLOCATION AND QOS ENHANCEMENTS OF IOT WITH FOG NETWORK." Journal of ISMAC, vol. 1, no. 2, pp. 21-30, 2019, doi: 10.36548/jismac.2019.2.003.

[12] G. Cui, X. Li, L. Xu, and W. Wang, "Latency and Energy Optimization for MEC Enhanced SAT-IoT Networks." IEEE Access, vol. 8, pp. 55915-55926, 2020, doi: 10.1109/access.2020.2982356.

[13] H. A. Alameddine, S. Sharafeddine, S. Sebbah, S. Ayoubi, and C. Assi, "Dynamic Task Offloading and Scheduling for Low-Latency IoT Services in Multi-

Access Edge Computing." IEEE Journal on Selected Areas in Communications, vol. 37, no. 3, pp. 668- 682, 2019, doi: 10.1109/jsac.2019.2894306.

[14] J. Liu and Q. Zhang, "Adaptive Task Partitioning at Local Device or Remote Edge Server for Offloading in MEC." 2020 IEEE Wireless Communications and Networking Conference (WCNC), 2020, doi: 10.1109/wcnc45663.2020.9120484.

[15] W. Almughalles, R. Chai, J. Lin, and A. Zubair, "Task Execution Latency Minimization- based Joint Computation Offloading and Cell Selection for MEC-Enabled HetNets." 2019 28th Wireless and Optical Communications Conference (WOCC), 2019, doi: 10.1109/wocc.2019.8770582.

[16] Y. Gu, W. Saad, M. Bennis, M. Debbah, and Z. Han, "Matching theory for future wireless networks: fundamentals and applications." IEEE Communications Magazine, vol. 53, no. 5, pp. 52-59, 2015, doi: 10.1109/mcom.2015.7105641.

[17] C. T. Do, N. H. Tran, C. Pham, M. G. R. Alam, J. H. Son, and C. S. Hong, "A proximal algorithm for joint resource allocation and minimizing carbon footprint in geo-distributed fog computing," in 2015 International Conference on Information Networking (ICOIN), 2015.

[18] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal workload allocation in fog- cloud computing towards balanced delay and power consumption," IEEE Internet Things J., pp. 1–1, 2016.

[19] D. Zeng, L. Gu, S. Guo, Z. Cheng, and S. Yu, "Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system," IEEE Trans. Comput., vol. 65, no. 12, pp. 3702–3712, 2016.

[20] H. Zhang, Y. Xiao, S. Bu, D. Niyato, F. R. Yu, and Z. Han, "Computing resource allocation in three-tier IoT fog networks: A joint optimization approach combining stackelberg game and matching," IEEE Internet Things J., vol. 4, no. 5, pp. 1204–1215, 2017.

[21] M. S. Elbamby, M. Bennis, and W. Saad, "Proactive edge computing in latency- constrained fog networks," in 2017 European Conference on Networks and Communications (EuCNC), 2017.

[22] T. Yang, R. Chai, and L. Zhang, "Latency optimization-based joint task offloading and scheduling for multi-user MEC system," in 2020 29th Wireless and Optical Communications Conference (WOCC), 2020.

[23] N. Nikaein, X. Vasilakos, and A. Huang, "LL-MEC: Enabling low latency edge applications," in 2018 IEEE 7th International Conference on Cloud Networking (CloudNet), 2018.

[24] B. Brik, P. A. Frangoudis, and A. Ksentini, "Service-oriented MEC applications placement in a federated edge cloud architecture," in ICC 2020 - 2020 IEEE International Conference on Communications (ICC), 2020.

[25] Alnoman, S. Erkucuk, and A. Anpalagan, "Sparse code multiple access-based edge computing for IoT systems," IEEE Internet Things J., vol. 6, no. 4, pp. 7152–7161, 2019.

[26] R. Han, Y. Wen, L. Bai, J. Liu, and J. Choi, "Rate splitting on mobile edge computing for UAV-aided IoT systems," IEEE Trans. Cogn. Commun. Netw., vol. 6, no. 4, pp. 1193– 1203, 2020.

[27] R. Gu, L. Yu, and J. Zhang, "MeFILL: A multi-edged framework for intelligent and low latency mobile IoT services," in 2020 IEEE Wireless Communications and Networking Conference (WCNC), 2020.

[28] Kovacevic, E. Harjula, S. Glisic, B. Lorenzo, and M. Ylianttila, "Cloud and edge computation offloading for latency limited services," IEEE Access, vol. 9, pp. 55764– 55776, 2021.

[29] H. Yoshino, K. Ota, and T. Hiraguri, "Adaptive control of statistical data aggregation to minimize latency in IoT gateway," in 2018 Global Information Infrastructure and Networking Symposium (GIIS), 2018.

[30] Chen, Y. Wang, Z. Fei, and X. Wang, "Power limited ultra-reliable and low-latency communication in UAV-enabled IoT networks," in 2020 IEEE Wireless Communications and Networking Conference (WCNC), 2020.

[31] J.-P. Hong, J. Park, W. Shin, and S. Beak, "Distributed antenna system design for ultra- reliable low-latency uplink communications," in 2019 International Conference on Electronics, Information, and Communication (ICEIC), 2019.

[32] M. Yang, S. Lim, S.-M. Oh, and J. Shin, "An Uplink Transmission Scheme for TSN Service in 5G Industrial IoT," in 2020 International Conference on Information and Communication Technology Convergence (ICTC), 2020.

[33] H. Ismail, T. A. Soliman, G. M. Salama, N. A. El-Bahnasawy, and H. F. A. Hamed, "Congestion-aware and energy-efficient MEC model with low latency for 5G," in 2019 7th International Japan-Africa Conference on Electronics, Communications, and Computations, (JAC-ECC), 2019.

[34] Zhang, S. Leng, Y. He, S. Maharjan, and Y. Zhang, “Mobile edge computing and networking for green and low-latency internet of things,” IEEE Commun. Mag., vol. 56, no. 5, pp. 39–45, 2018.

[35] J. Zhang, X. Xu, K. Zhang, S. Han, X. Tao, and P. Zhang, “Learning based flexible cross- layer optimization for ultra-reliable and low latency applications in IoT scenarios,” IEEE Internet Things J., pp. 1–1, 2021.

[36] N. Germenis, P. Fountas, and C. Koulamas, “Low latency and low cost smart embedded seismograph for early warning IoT applications,” in 2020 9th Mediterranean Conference on Embedded Computing (MECO), 2020.

[37] G. Calice, A. Mtibaa, R. Beraldi, and H. Alnuweiri, “Mobile-to-mobile opportunistic task splitting and offloading,” in 2015 IEEE 11th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), 2015.

[38] N. Kherraf, S. Sharafeddine, C. M. Assi, and A. Ghrayeb, “Latency and reliability-aware workload assignment in IoT networks with mobile edge clouds,” IEEE trans. netw. serv. manag., vol. 16, no. 4, pp. 1435–1449, 2019.

[39] H. Hao, Y. Wang, Y. Shi, Z. Li, Y. Wu, and C. Li, “IoT-G: A low-latency and high- reliability private power wireless communication architecture for smart grid,” in 2019 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm), 2019.

[40] J. Park and Y. Lim, “Balancing loads among MEC servers by task redirection to enhance the resource efficiency of MEC systems,” Appl. Sci. (Basel), vol. 11, no. 16, p. 7589, 2021.

[41] SFogSim. [Online]. Available: https://github.com/ saifulislamPhD/SFogSim

[42] Parallel Workload Archive. http://www.cs.huji.ac.il/labs/parallel/workload/

[43] “Experience with using the Parallel Workloads Archive” (J. Parallel & Distributed Comput. 74(10) pp. 2967-2982, Oct 2014)

Dr. Sandeep Kumar Nayak
Dr. Sandeep Kumar Nayak received his bachelor degree in Computer Science from Lucknow University and Masters in Computer Application from UP Technical University, Uttar Pradesh, and Lucknow, India. He received his PhD from Babasaheb Bhimrao Ambedkar University, Lucknow (A Central University of India). He is currently an Associate Professor in Department of Information Technology, Babasaheb Bhimrao Ambedkar University- A Central University, Amethi, India. His research interest is on a variety of topics like Big Data, Software Engineering, Internet of Things, Sensor Networks, Cyber Forensics, Crowdsourcing, Data Science and Cloud Computing.

Ms. Eram Fatima Siddiqui
Ms. Eram Fatima Siddiqui received her bachelor degree in Computer Application and Masters in Computer Application from Integral University, Uttar Pradesh, and Lucknow, India. She is currently a Research Scholar in Department of Computer application, Integral University, Lucknow, India. Her research topic is based on a collaborative approach of Internet of Things and Fog computing. Her interest is on a variety of topics like Big Data, Edge Computing, Data Science, Machine Learning, Blockchain, Internet of Things, Sensor Networks and Cloud Computing.