IoT-Orchestration based Nanogrid Energy Management System and Optimal Time-Aware Scheduling for Efficient Energy Usage in Nanogrid

Faiza Qayyum[1], Harun Jamil[2], Faisal Jamil[3], Shabir Ahmed[3], Do-Hyeun Kim[1 *]

[1]Department of Computer Engineering (Research Center of Advance Technology), Jeju National University, Jeju National University, Republic of Korea

[2]Department of Electronic Engineering, Jeju National University, Republic of Korea

[3]Department of IT Convergence Engineering, Gachon University, Sujeong-Gu, Seongnam-Si, Gyeonggi-Do, 461-701, Korea

*Corresponding Author: Faiza Qayyum. Email: faizaqayyum@jejunu.ac.kr

Abstract:

The present era of the Internet of Things (IoT) having intelligent functionalities in solving problems pertaining to real-time mission-critical systems has brought an immense revolution in diverse fields including healthcare and navigation systems. However, to the best of our knowledge, the potential of IoT has not been fully exploited yet in the field of the energy sector. We argue that there is an immense need to shift the traditional mission-critical electric power system architecture to IoT-based fully orchestrated architecture in order to increase efficiency, as billions of investment is reserved for the energy sector globally. Since network orchestration deals with auomating the interaction between multiple components involved to execute a particular service, therefore, scheduling the relevant processes within strict deadlines becomes the core pillar of the architecture. The mission-critical systems with urgent task execution often suffer from issues of missing task deadlines. In this study, we present a novel IoT task orchestration architecture for efficient energy management of a nanogrid system that focuses on minimizing the use of non-renewable energy resources and maximizing the use of renewable energy resources. Moreover, major components of IoT task orchestration such as task mapping and task scheduling are also enhanced using NLP and PSO optimization modules. The proposed task scheduling algorithm incorporates the optimized surplus time, and efficiently executes the energy management-related tasks contemplating to their types. The study utilizes sensors to obtain data from physical IoT devices, including photovoltaic (PV), Energy Storage System (ESS), and diesel generator (DG). The performance of the proposed model is evaluated using data set of nanogrid houses. The outcomes revealed that IoT-task orchestration has played a pivotal role in efficient energy management for nanogrid mission-critical system. Furthermore, the comparison with state-of-the-art scheduling algorithms showed that the task starvation rate is reduced to 16% and 12% when compared with RR and FEF algorithms, respectively.

Keywords: Internet of things; complex problem solving; Critical IoT systems; Microgrid; Nanogrid; Optimization; Scheduling; Task modeling; Task orchestration;

1. Introduction

1.1 Background

Over the past few years, the Internet of Things (IoT) is deemed to have revolutionized multifarious disciplines encompassing smart homes, smart cities, and healthcare applications [1-4]. Ashton Kevin is among the pioneers of introducing IoT technology, which emphasizes transforming the physical world into a digital world to execute the relevant operations smartly [5-7]. IoT comprises a network containing physical devices embedded with sensors, software, or other technologies, to have seamless communication among them or with humans over the Internet. The devices range from sophisticated industrial tools to ordinary household objects [8].

An analysis of the unprecedented growth of IoT technology shows that the number of IoT-connected devices will exceed 25.4 billion by 2030, wherein 60% of the ratio belongs to mission-critical applications [9]-[11]. Some of the examples of mission-critical systems include real-time navigation systems, healthcare systems, and enegy power systems [4, 5, 6,7, 12, 13] . International energy agency [IEA] revealed that the energy sector is expected to invest 40 trillion USD by 2030 [14]. An energy management system, responsible for ensuring real-time power delivery to electrical applications amid power outages is termed as a mission-critical electric power system [15].

In 2002, R. H. Lasseter introduced microgrids with the notion of a low-voltage and eco-friendly distribution system that simultaneously controls distributed generations systems [16]. A nanogrid is a simpler form of a microgrid that involves a single load encompassing a single house [17]. In other words, NG is deemed a single entity responsible for administration, voltage, and reliability, and having at least gateway at the outside [18]. A microgrid or nanogrid installed within a mission-critical entity, such as a hospital or healthcare clinic, wherein the process of prompt energy provision must not be compromised, fall in mission-critical systems. One of the potential reasons for microgrids/nanogrids usage is their reliance on renewable energy resources (RES), such as, photovoltaic (PV) and wind turbine (WT), and capacity to operate in both grid-connected and Island mode [15], [18]. The inherited disadvantages of RES are addressed using hybrid renewable energy resources wherein both RES and non-RES are utilized to fulfill energy requirements. The scientific community has presented varying methods to determine the best use of both the resources, i.e., maximizing RES and minimizing non-RES (diesel), and minimizing cost by optimizing energy management systems for microgrids and nanogrids [19, 21]. Similar to mission-critical systems in healthcare [4], the performance of mission-critical microgrid or nanogrid systems can significantly be enhanced by leveraging IoT technology in optimal energy management systems. One example of such a system is [4], wherein IoT task orchestration is applied to improve the perfromance of mission-critical healthcare systems.

1.2 Motivation

As explained earlier, a vast number of mission-critical systems from healthcare and navigation domains have recently been elevated with IoT technology by replacing the manual operating process with fully automated systems using edge or fog based IoT technologies. However, minimal attention has been paid to leveraging IoT technology for electric power management, such as microgrid [15] or nanogrid energy management systems [18]. An analysis shows that the building sector consumers consume around 40% of the global energy [22]. Intelligent integration of IoT orchestration with nanogrid energy management systems can help manage the entire system in a virtualized mode, enhancing the overall efficiency [5, 6]. We believe that leveraging IoT technology can immensely escalate the performance of electric power systems. Orchestrating the architecture of different mission-critical systems has been proven quite helpful in autonomous process handling [23]-[24].

Orchestration is a process that provides automated configuration, management, and coordination of IoT-enabled system components, especially in scenarios requiring real-time response [5, 6]. Unlike other domains, especially the healthcare sector, smart grid-based systems still need to be orchestrated to fully exploit their potential in terms of scalability and flexibility via autonomous energy operations. The contemporary state-of-the-art has introduced service-level orchestration, but very few have focused on implementing the orchestration at the task level [5].

Furthermore, mission-critical systems suffer from inevitable delays, adversely impacting the overall system's execution [5]-[7]. For instance, consider a scenario wherein energy provision must never be interrupted, i.e., a hospital relying on a microgrid power system or a nanogrid installed in a small-scale healthcare center performs an emergent process/job that requires instant power. Lack of real-time job execution ability may halt the overall system execution resulting in losing consumer reliance. The IoT task execution follows a workflow with an end-to-end deadline in which an output of one task serves as an input for the followed task. In such a system, it is preferred for a job to produce approximate results than an overdue precise result [25]. Multiple studies have emphasized improving the orchestration and scheduling of such workloads in fog computing frameworks [22]. Since scheduling in such a systems is NP-hard problem, therefore, heuristic models are harnessed for optimize scheduling. Hence, enabling IoT orchestration with nanogrid enegry management system and optimal scheduling of the jobs involved in the orchestration are primary concerns of this study.

1.3 Contribution

To this end, firstly, this study introduces IoT task orchestration architecture to implement nanogrid energy management system [15], in a fully virtualized manner, which escalates scalability and efficiency of the overall system. The physical devices used for energy management, such as, PV, WT, diesel generator (DG), and energy storage system (ESS) are replicated with virtual objects assuming all the properties of the devices. The processes/jobs related to energy management are automatically mapped to their targetted devices with the help of IoT sensors and raspberry PI. Secondly, optimal time-aware task scheduling algorithm incorporating an objective function is also presented. The algorithm focuses on determining the best execution order of the jobs during orchestration, maximizing the response time, and minimizing the task idle time.

To recapitulate, the main contributions of the proposed study are listed below:

- Implements virtual nanogrid energy management architecture harnessing real-time IoT task orchestration where tasks are dynamically generated and automatically assigned to the corresponding physical device to increase efficiency and robustness;
- Designed PSO-enabled objective function to optimize the surplus time of the tasks and utilize it as an input to the proposed scheduling technique designed for the core part of IoT task orchestration (i.e., scheduling);
- Developed time-aware optimal task scheduling algorithm that considers optimized task surplus time to determine the best execution order of the real-time tasks involved in the orchestration of the energy management;
- A case study encompassing data of 24 nanogrid houses having PV, WT, DG, and ESS installed, is employed to evaluate the performance of the proposed architecture.
- Standard evaluation measures, round trip time (RTT), response time, throughput, latency, task drop, and starvation rate are utilized to assess the potential of the proposed scheduling technique;
- A comparison is also drawn with one of the existing studies to specify the standing of the proposed study in contemporary state-of-the-art.

The remainder of the paper is structured as follows. Section 2 presents the literature review and highlights the relevant contributions in mission-critical IoT architecture. Section 3 illustrates the system model for the architecture, which designs an optimized orchestration mechanism. Section 4 respresents the result attained using simulation of data of houses and section 5 concludes the paper.

2. Related work

This section discusses the existing state-of-the-art related to orchestration in IoT based systems and optimal scheduling method proposed to improve the task execution process of different services.

It is essential to be aware of the deterministic nature of mission-critical operations such as real-time energy provision in smart grids [26]. IoT-enabled mission-critical applications demand high accuracy as well as a fast response before employing in a real environment. The revolution of IoT has enriched the ubiquity of smart sensor-based devices [27, 28], and escalated the performance of applications pertaining to multiple domains, including smart grid [29], smart cities [30], healthcare systems [4], and enterprise systems [31,32]. The innovation in IoT technology has paved the way toward the efficient architecture of IoT applications. Orchestration is deemed a vital concept in service-oriented architecture in terms of design [33, 34]. It entertains the concept of automating the entire execution process. The orchestration has grabbed the major attention of the scientific community when implemented in SOA [35, 36]. It is considered an essential component in conventional and to-date IoT applications. The study [37] presented a "IoT ProSe" model that implements the task orchestration in mobile nodes. ProFun [38], Makesense [39] suggest task-level programming to optimize the efficiency; however, this study focuses more on the dynamic aspect than autonomy for the orchestration.

Healthcare and other domains have fully exploited the potential of IoT technology and fog systems by implementing orchestration [4, 22]. In [6], authors have presented multi-device multi-tasks management (MDMT-MOA) and orchestration architecture for IoT enterprise frameworks. Another study [40] presents complex problem solutions as a service (CPSaaS) using optimal task orchestration for smart city. In [41] a mission-critical system is implemented to detect mountain fire using IoT task orchestration that contains microservice and predictive analysis as core modules. Similarly, in healthcare domains, the studies [42, 43] employ sensors to detect the motion of a patient in IoT health monitoring systems. In [44], fog and cloud computing-based health monitoring framework is presented, resulting in efficient healthcare data management. The study [4] presented an IoT-tasks orchestration to monitor patient health monitoring, an optimal scheduling algorithm for healthcare tasks execution. The study follows task generation (tasks related to the desired service are generated), task mapping (tasks are mapped to the virtual objects created for physical devices), Task scheduling (schedule the mapped task), task allocation (tasks are allocated to physical devices), task deployment (tasks are deployed to physical devices) based hierarchy of IoT orchestration which resulted in significantly improved response time and minimized latency. In addition, contemporary state-of-the-art on orchestration focuses on automating the service provision process. However, there exist numbered studies that implement task-level orchestration. Task-level orchestration may enhance flexibility, robustness, and dynamic handling of the overall system execution [5-7].

Task scheduling serves as a core module of IoT task orchestration; therefore, its optimization can positively impact the overall system's performance. In an ideal scheduling, a task gets a fair chance of execution by ensuring the resource does not remain idle for a long-time span. A few studies focus on overcoming real-time IoT systems' scheduling-related deficiencies [45-49]. A varying studies presented varying task scheduling techniques to improve the process execution in multiple domains [50-55]. In [50], a resource allocator and IoT-based service delegation model is presented to control users' requests on suitable cloud/fog efficiently. The study [51] has propsoed predict earliest finish time (PEFT) algorithm for job scheduling.

Similarly, multiple workflows in a cloud computing environment are scheduled using a clustering-based scheduling algorithm [52]. The study [53] suggests a novel method for optimal resource allocation and fault tolerance and minimizes resource overflow using an efficient resource allocator (ERA). The study [54] presented a novel scheduling method to improve the quality of service (QoS) to attain execution trade-off among applications' performance. Data mining-based task scheduling was proposed in [55] to schedule the tasks in fog computing systems efficiently. One of the recent studies [4], presented time-constraint aware task scheduling to optimize the execution of healthcare tasks related to orchestrated mission-critical patient monitoring IoT framework. In an ideal scheduling, multiple processes get a fair chance of execution by ensuring the resource does not remain idle for a long-time span.

Critical analysis of the contemporary state-of-the-art revealed a dire need to exploit IoT technology in the energy sector fully. To this end, similar to other domains, distributed generation-based energy management systems, such as nanogrids or microgrids, can significantly be improved by employing IoT-task orchestration. In addition, the scheduling module of the IoT task orchestration can also be enhanced considering major aspects overlooked by the contemporary state-of-the-art.

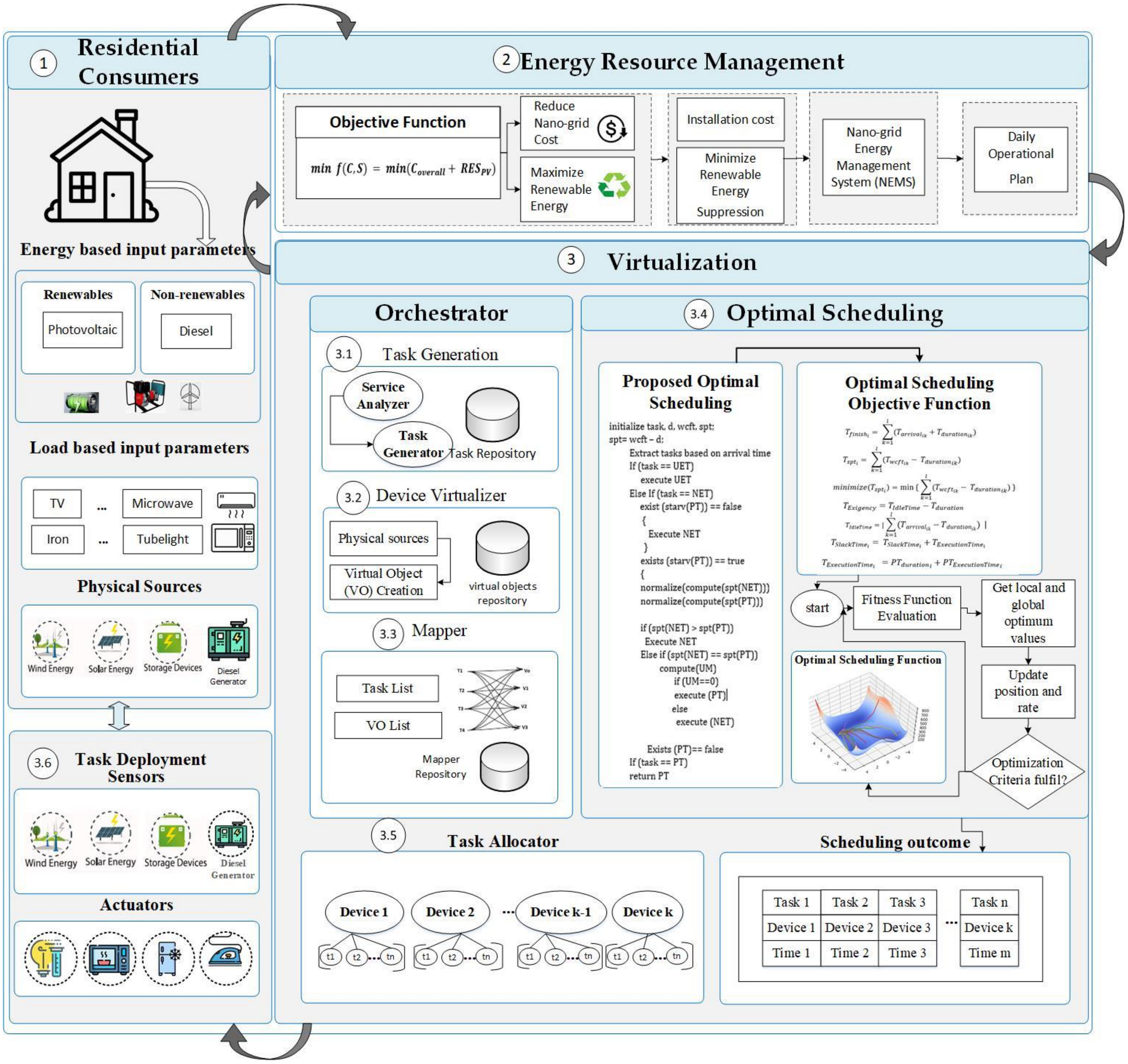
3. Methodology

This section encompasses the methods and materials employed to implement the proposed time-stamp-based optimal scheduling mechanism. The overall architecture of the proposed methodology is envisioned in Figure 1. Firstly, input parameters of 24 smart nano-grid houses, such as photovoltaic (PV), wind turbine (WT), diesel generator (DG), ESS capacity, and load demands, in KwH are considered on daily basis. These input parameters are forwarded to the energy resource management module, which implements the energy management system presented by [38]. Then, the tasks required to implement the energy management algorithm are given to the virtualization module, which generates the task according to the requirement of the energy management system. The tasks are generated using the NLP-based service analyzer. Afterward, virtual objects are created by replicating the physical resources. Next, the tasks and virtual objects (VOs) are mapped using the proposed NLP-based method. The physical IoT resources contain the installed physical devices like sensors and actuators. The sensors sense and transfer the energy-related requirements' data to IoT servers. At the hardware end, we use Raspberry-pi with which four sensors are embedded to obtain data from the physical resources; for instance, a separate sensor is used to obtain data from PV, WT, ESS and DG. Though the hardware implementation is pretty simple; however, adequate to acquire the optimal orchestration for real-time based mission-critical applications. In simple words, this study includes three basic components; task management server using PC, Edge node based on Raspberry PI and embedded devices associated with sensors and actuators and is also based on Raspberry PI.

3.1 Design details of the proposed model

This subsection delineates the architecture of proposed optimal scheduling, of which the architecture is envisioned in Figure 1. The proposed architecture follows a six-layered top-bottom approach that implements nano-grid energy resource management using a task orchestration framework. The study considers the scenario of smart grid-based mission critical applications wherein a consumer requires instant power in an optimal and cost-efficient manner. Contemplate a scenario that requires a prompt response for a real-time mission-critical system, i.e., a hospital relying on a microgrid power system or a small-scale health-center depending on nanogrid-based power system, performs some necessary procedure. In such a situation, the process of instant power provision must not be compromised as the missing task deadline can cause severe consequences. This study presents an enhanced optimal task scheduling algorithm that uses task surplus time and optimizes it using an objective function. It then incorporates it as a deciding factor to determine the best execution order of the energy tasks. The objective of the proposed scheduling method is to have a minimal chance of missing the tasks' deadline. A typical flow of task orchestration spans the following steps: (1) task generation, (2) device virtualization, (3) task mapping, (4) task scheduling, (5) task allocation and (6) task deployment. First, the task generation analyzes the service description and decomposes it into multiple tasks using the natural language processing (NLP) method [12]. Secondly, device virtualization step creates virtual objects by replicating the physical IoT resources. The resources are photovoltaic (PV), wind turbine (WT), diesel generator (DG), and energy storage system (ESS) as we consider a nano-grid-based smart energy management system.

Next, the generated tasks are mapped to their corresponding virtual objects (VOs) using our proposed NLP-based cross-mapping that considers part-of-speech (PoS) to determine the relationship between tasks and VOs.



```
initialize task, d, wcft, spt;
spt= wcft – d;
    Extract tasks based on arrival time
    If (task == UET)
      execute UET
    Else If (task == NET)
      exist (starv(PT)) == false
      {
        Execute NET
      }
      exists (starv(PT)) == true
      {
      normalize(compute(spt(NET)))
      normalize(compute(spt(PT)))

      if (spt(NET) > spt(PT))
        Execute NET
      Else if (spt(NET) == spt(PT))
           compute(UM)
              if (UM==0)
              execute (PT)
             else
              execute (NET)

        Exists (PT)== false
    If (task == PT)
    return PT
```

$$T_{finish_i} = \sum_{k=1}^{l}(T_{arrival_{ik}} + T_{duration_{ik}})$$

$$T_{spt_i} = \sum_{k=1}^{l}(T_{wcft_{ik}} - T_{duration_{ik}})$$

$$minimize(T_{spt_i}) = \min\{\sum_{k=1}^{l}(T_{wcft_{ik}} - T_{duration_{ik}})\}$$

$$T_{Exigency} = T_{IdleTime} - T_{duration}$$

$$T_{IdleTime} = |\sum_{k=1}^{l}(T_{arrival_{ik}} - T_{duration_{ik}})|$$

$$T_{SlackTime_i} = T_{SlackTime_i} + T_{ExecutionTime_i}$$

$$T_{ExecutionTime_i} = PT_{duration_i} + PT_{ExecutionTime_i}$$

Task 1	Task 2	Task 3	...	Task n
Device 1	Device 2	Device 3	...	Device k
Time 1	Time 2	Time 3		Time m

Figure 1 Optimal IoT task orchestration architecture and Optimal Scheduling Algorithm

A contemporary task orchestration model adopts the manual drag-drop method for task mapping; however, we diminish this manual effort by using autonomous mapping. The fourth step is the core module of the proposed model wherein task tuples, comprising tasks, virtual objects and timestamp at which the mapping process was performed, i.e., (T, VO, t) are processed to determine the task execution order using the proposed scheduling technique. Finally, the task allocation phase allocates the tasks to the corresponding sensors, and at the end, the tasks are deployed to the targeted physical devices. The steps are detailed explained in the following sub-sections.

3.2 Nano-grid Energy Management System

The employed energy management algorithm is well presented in [15] and [18]. To demonstrate a clear linkage of the energy management framework in context for this study, we explain the applied Nanogrid Energy Management System (NEMS) with the help of the flowchart shown in Figure 2. The focus of NEMS is to reduce nano-grid cost and minimize suppression of renewable energy sources. NEMS takes home appliances load data, weather data, PV, DG, and ESS data as input. The parameters are given to the objective function, which focuses on reducing nano-grid installation, operational, and maintenance costs by using the required data from appliances load, renewable and non-renewable energy resources. The objective function is implemented using PSO. The values produced by the objective function serve as input to the NEMS to minimize the suppression of renewable energy resources. The NEMS considers battery state-of-charge (SoC), the value of which serves as a threshold to minimize or maximize the usage of diesel, PV, WT to the certain level. For instance, SoC greater than 0.9 is considered as overcharged. If SoC is between 0.8 and 0.9, ESS is considered adequate charged. If SoC is less than 0.1 and greater thanb y0, then DG amount is fixed to a certain level. In EMS, y0 is the ouput value received by the first level PSO optimizer, used as a decision variable, explained in detail in [15]. Similarly, if SoC is 0.1, ESS is considered undercharged and less than 0 empty ESS. This study implements the entire energy management process using task orchestration. As explained earlier, to the best of our knowledge, this is the first attempt that involves task orchestration in energy resources management, as shown in Figure 2.

3.3 Task Orchestration Architecture

This section presents a detailed explanation of each step of the orchestration. Figure 3 shows a detailed orchestration framework containing (1) Task generation using server on PC, (2) Task mapping, (3) Energy task scheduling using the proposed optimal scheduling method, and (4) Task allocation and deployment.

3.3.1 Task generation using server on PC

Task generation provides feasibility to users in terms of delivering service titles and descriptions using a statement encompassing words of their choice specifying the current requirement for energy management. The module is responsible for analyzing and decomposing the services into further microservices (i.e., tasks) with the help of NLP technique [8] embedded with the service analyzer so that both the explicit and implicit meaning of the provided description can be ascertained. NLP-based mechanisms are widely employed in text mining; wherein text is required to be processed to ensure maximum accuracy of an information retrieval system [39]. The relevant NLP techniques include tokenization, part of speech (PoS) tagging, text similarity, semantic similarity, etc. The intention of these approaches is the automatic interpretation of meaning in human languages. Therefore, we also employ the NLP module of [8] to implement the task orchestration. Once the NLP generates the input tasks related to nano-grid energy management, the micro-service analyzer forwards them to the task generator manager (TGM), which store them in the task repository. The tasks are segregated into two categories: (1) periodic and (2) event-based tasks, using adverb followed by the main verb of the description. For instance, the following tasks are generated automatically using the above-mentioned procedure, *get solar get wind, get ESS* and *get diesel.* The tasks used to transfer sensing data from sensing devices to physical devices using IoT server, such as reportSolar, reportWind, reportESS, and reportDiesel. The purpose of these tasks is to get the real-time energy values of four physical resources, i.e., photovoltaic (PV), wind turbine (WT), energy storage system (ESS), and diesel generator (DG) in KwH.

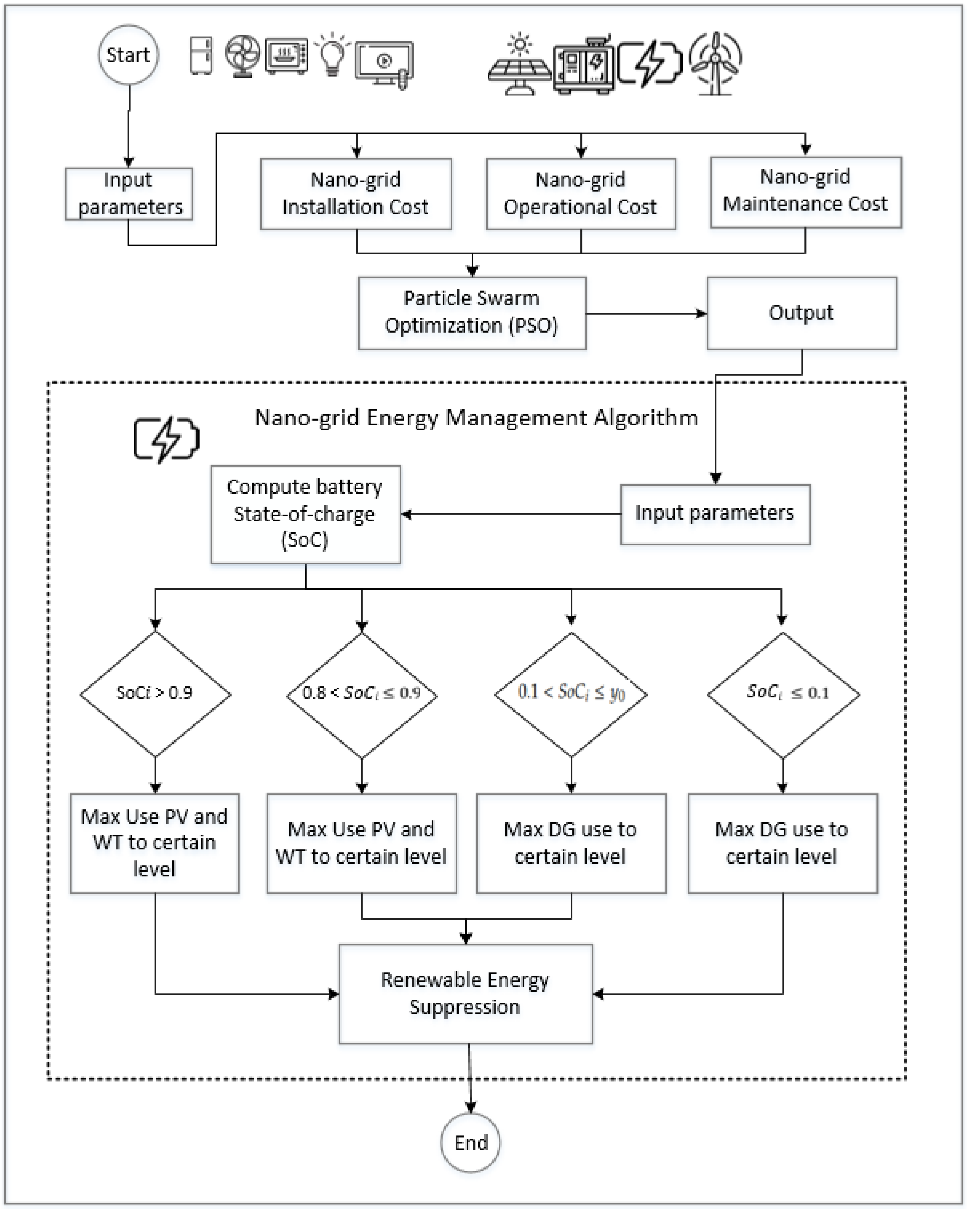

Start
Input parameters
Nano-grid Installation Cost
Nano-grid Operational Cost
Nano-grid Maintenance Cost
Particle Swarm Optimization (PSO)
Output
Nano-grid Energy Management Algorithm
Compute battery State-of-charge (SoC)
Input parameters
SoCi > 0.9
0.8 < SoC_i ≤ 0.9
0.1 < SoC_i ≤ y0
SoC_i ≤ 0.1
Max Use PV and WT to certain level
Max Use PV and WT to certain level
Max DG use to certain level
Max DG use to certain level
Renewable Energy Suppression
End

Figure 2 Nano-grid Energy Resource Management System

3.3.2 Device virtualization (VO Creation)

This module generates virtual objects by replicating the IoT resources such as PV, WT, ESS and DG. The virtual resource manager (VRM) creates these objects and stipulates the interface to apply CURD operations on the virtual objects. The created VOs hold all the physical resources' characteristics and additional characteristics by the IoT virtualization environment. The VOs have the functionality according to the respective physical resource and other

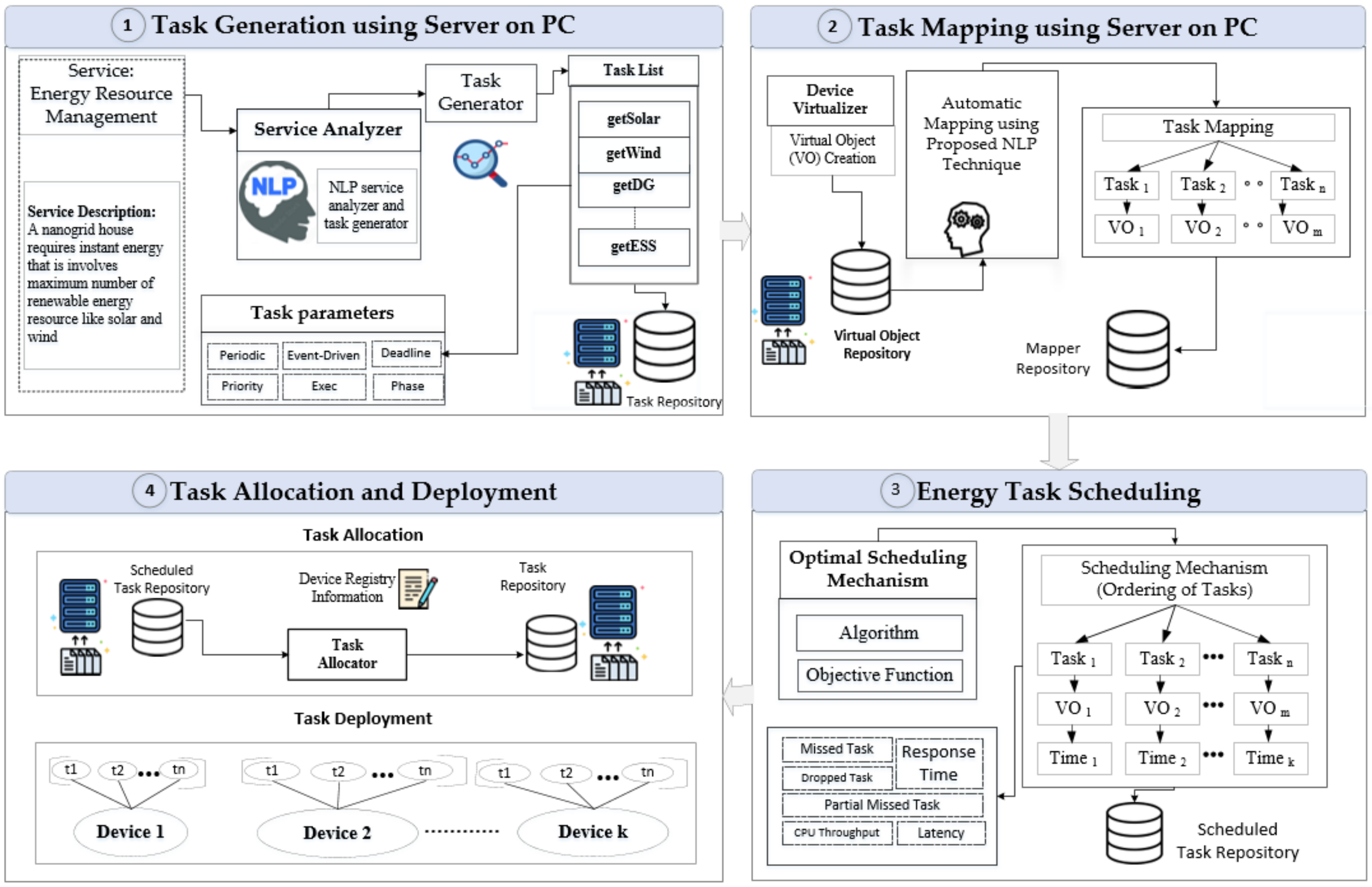

1 Task Generation using Server on PC
Service: Energy Resource Management
Service Description: A nanogrid house requires instant energy that is involves maximum number of renewable energy resource like solar and wind
Service Analyzer
NLP
NLP service analyzer and task generator
Task Generator
Task List
getSolar
getWind
getDG
getESS
Task parameters
Periodic
Event-Driven
Deadline
Priority
Exec
Phase
Task Repository
2 Task Mapping using Server on PC
Device Virtualizer
Virtual Object (VO) Creation
Automatic Mapping using Proposed NLP Technique
Task Mapping
Task 1
Task 2
Task n
VO 1
VO 2
VO m
Virtual Object Repository
Mapper Repository
3 Energy Task Scheduling
Optimal Scheduling Mechanism
Algorithm
Objective Function
Scheduling Mechanism (Ordering of Tasks)
Time 1
Time 2
Time k
Missed Task
Response Time
Dropped Task
Partial Missed Task
CPU Throughput
Latency
Scheduled Task Repository
4 Task Allocation and Deployment
Task Allocation
Scheduled Task Repository
Device Registry Information
Task Allocator
Task Repository
Task Deployment
t1
t2
tn
Device 1
Device 2
Device k

Figure 3 Hierarchical layout of IoT Task Orchestration for Nanogrid Energy Management

relevant information, including location and URI. The VRM also shows the graphical representation of virtual objects. In the end, the created virtual objects are stored in the virtual object repository.

3.3.3 Task Mapping to Virtual Objects

The third step of the orchestration process is to map the generated tasks into the corresponding virtual objects. For instance, a task "getSolar" has to be executed by the device "photovoltaic (PV)", so in the proposed architecture, the task will be mapped to the virtual object representing the device. The generated tasks are extracted from the task repository and virtual objects are extracted from the VO repository. The existing orchestration-based studies adopt a manual drag and drop procedure to map the tasks to corresponding virtual objects; however, this incorporates manual effort in the orchestration framework. We present NLP-based automatic task mapping that mitigates the need of manual mapping. We apply the cosine similarity measure and string-matching approach of [39], which computes the similarity between the generated task and virtual objects. If the similarity exceeds more than 70% then it is ensured that the task belongs to the virtual object. A task may be mapped to more than one virtual object. For instance, the tasks "getsolar" and "reportsolar" will be mapped to the VO representing photovoltaic. These pairs collectively form a Uniform Resource Identifier (URI) that holds information to analyze the output of the specific process. The URI is also harnessed to get access the physical IoT resource. The task mapping controller (TMC) calls the proposed NLP-based mapping procedure to link the task list and physical resources. The TMC graphically represents the mapped tasks and is also used to store mapping configurations for further processing. The task mapping holds the following parameters: (1) task ID, (2) virtual object ID, and (3) timespan at which the particular pair is formed.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 10 February 2022 doi:10.20944/preprints202202.0150.v1

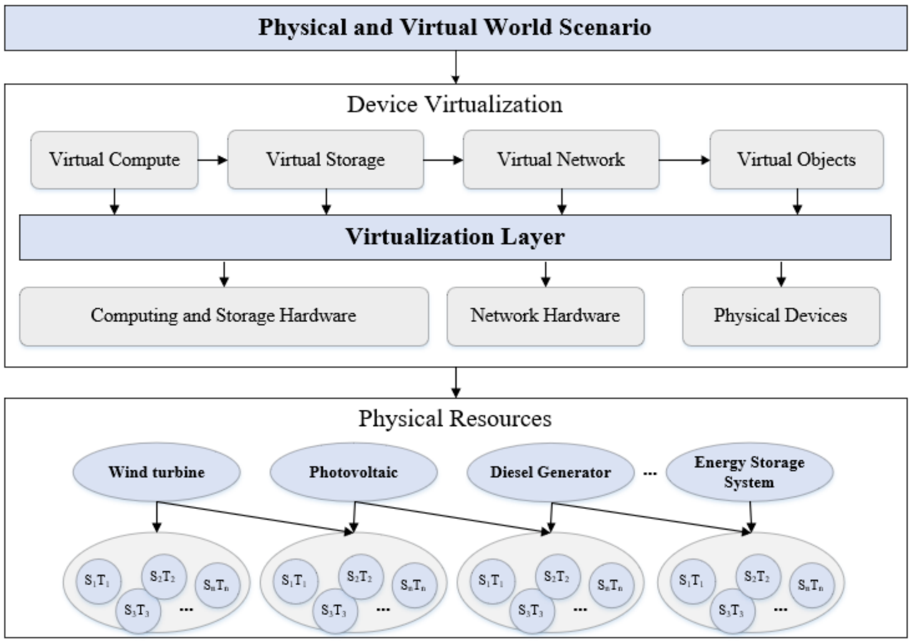


Figure 4 Physical and virtual world scenario in proposed Task Orchestration

In other words, the task mapping module takes generated tasks and virtual objects as an input, applies the proposed NLP technique for their mapping and produces output in the form of a message having the following attributes: Task ID, which identifies the current task, VO ID, that identifies the virtual object, Message-ID which uniquely identifies the message, Microservice ID, which identifies the relevant microservice task and timestamp, providing information about the time interval at which the certain mapping was done. In the end, a tuple comprises tasks, virtual objects, and time-stamp at which the mapping process was performed, i.e., (T, VO, t) is forwarded to the scheduling module.

3.3.4 Proposed Hybrid Optimal Task Schedular

The contemporary scheduling algorithms do not produce promising results when addressing real-time problems mission-critical systems encounter. In these systems, tasks usually have absolute/dynamic deadlines. For example, a real-time system might consist of event-driven tasks, periodic tasks, or a combination of both. Periodic tasks have regular arrival times, while event-driven tasks have irregular arrival times. The objective of real-time task scheduling is to determine the best execution order of the tasks. Scheduling is performed by a module called scheduler, which mainly aims to maximize the throughput and fairness while minimizing the tasks' average waiting and response time. When implementing a scheduler, it becomes tough to ensure all these requirements at a single time. If the real-time system is complex, then the task’s deadline becomes one of the most important factors, as a task must be completed before its deadline. Other priorities might include a task’s period, arrival time, slack, or any user-defined priority. Hence, task scheduling mainly focuses on scheduling the tasks based on their priorities in the real-time systems. In priority-based scheduling policies, an important concept is to avoid task starvation. Starvation can be defined as preventing a task from completing its execution for a long time because the resources are allocated to some other tasks. The best scheduler is one, which also competes to save the lower priority tasks from starvation and take care of all the priorities. Hence, we can conclude that the primary objective of a scheduling algorithm is to decide the order in which tasks are to be executed so that the minimum number of tasks is deprived. At the same time, the maximum of the defined goal is achieved.

3.3.5 Systematic Flow of proposed Optimal Schedular

IoT devices often have limited resources and require control tasks to be executed within milliseconds. Figure 5 shows the task scheduling. The prime focus of this study is on an effective scheduling mechanism to schedule the energy tasks on physical IoT devices. The correct execution order is determined by the task scheduling manager (TSM). A task holds characteristics including arrival time, execution time, deadline time, priority, execution urge (i.e., soft/hard), a control task, sensing information, to decide execution order for the tasks.

A candidate task is forwarded to TSM in a tuple (i.e., T, VO, t) for execution. TSM fetches these attributes from the respective repositories. For instance, tasks are fetched from the task repository, virtual objects are fetched from the VOs repository, and output is produced by the task mapping controller (TMC) to commence the scheduling method employing the proposed scheduling technique. This study presents an optimal time-stamp-based scheduling technique to have an optimal execution plan. It considers the essential aspects overlooked by the contemporary scheduling algorithms, such as task surplus time, priorities, and other essential aspects such as existing starving tasks and missing and delay ratio. The proposed algorithm focuses on optimizing the task surplus time to maximize the performance of the proposed algorithm.

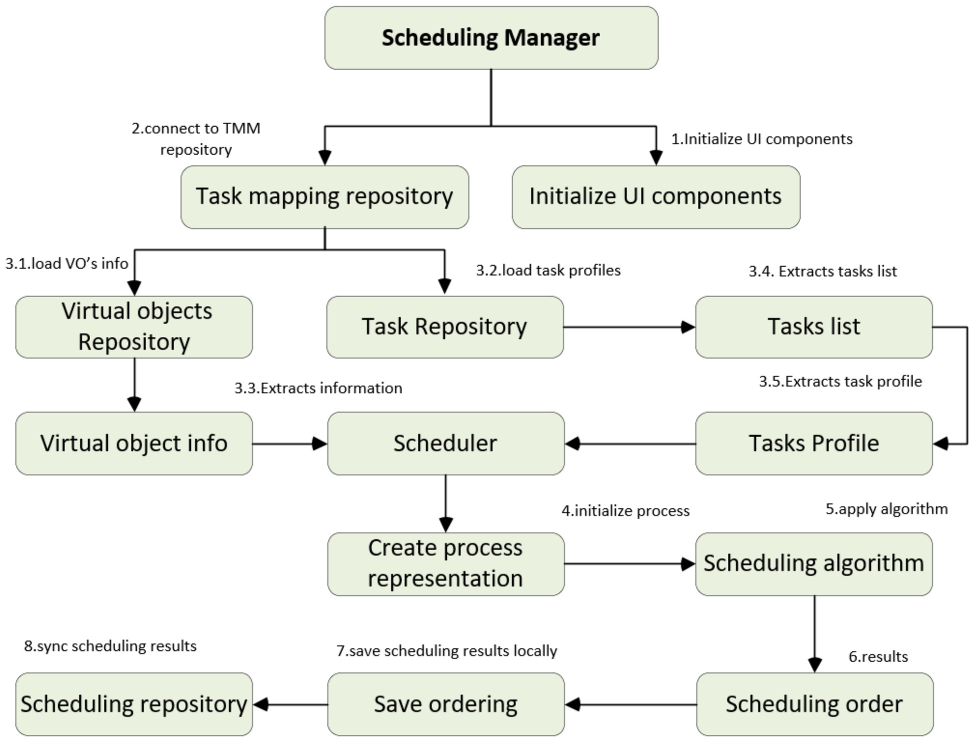


Figure 5 Systematic flow of proposed Optimal Schedular

The surplus time specifies a duration for which a particular task can wait. The surplus time is optimized using the proposed objective function implemented using the particle swarm optimization (PSO) algorithm. The focus of PSO is to reduce the surplus time so that an optimal decision can be taken while selecting the tasks for execution. PSO is deemed one of the renowned optimization algorithms that adopt the flying pattern of the birds' swarm in search of food [52]. First, the local and global best positions of particles are determined by evaluating each particle using the proposed objective function. Next, the velocities are updated by harnessing the pbest and gbest values to reach optimal solutions. Afterward, the particle's direction and speed are calculated using the velocity vector. Lastly, the position of each particle is updated to obtain the global optima.

Typcially, a smart grid-based system comprises two types of tasks: 1) periodic tasks and 2) event-driven tasks. Contemplating on the nature of both of the tasks, the proposed algorithm segeregates them into two types: (1) Periodic tasks into *priority periodic task (PPT)*, *simple periodic task (SPT)*, and (2) event-driven tasks into *prompt event-driven tasks* and simple *event-driven task (SET)*. Inefficient task management and failing to meet the electricity demands on a real-term basis can harm the entire smart grid system. Therefore, optimal scheduling of the tasks per the requirements on real grounds can significantly boost the performance. In this regard, a PSO-based optimal scheduling mechanism is proposed to schedule the hard and soft tasks effectively and dynamically. Table 1 recapitulates the notations used in the equations and proposed objective function.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 10 February 2022 doi:10.20944/preprints202202.0150.v1

Table 1 Notations Explanation

Name	**Definition**
Sensing Devices (D_i)	$D_i = \{D_1, D_2, D_3, \ldots, D_n\}$, n number of devices
Tasks	Tasks $T_i = \{T_1, T_2, T_3, \ldots, T_m\}$, m number of sensing devices
Surplus time (spt)	spt= surplus time, which determines that how long a task can wait in the queue
Task identifier (T_{id})	T_{id}: Identifier of the smart grid-based task
Virtual object identifier (VO_{id})	VO_{id} : Identifier of a virtual object
Task arrival time ($T_{arrival}$)	$T_{arrival}$: The time at which the task arrives
Task finish time (T_{Finish})	T_{Finish} : The time at which the task finishes its execution
Task priority ($T_{Priority}$)	$T_{Priority:}$ defines priority of a task
Task duration ($T_{duration}$)	$T_{duration}$:the timespan consumed by a task to complete the execution
Task exigency ($T_{Exigency}$)	$T_{Exigency}$: Exigency measure that determines urgency of the tasks in case when decision cannot be taken based on surplus time.
Task Failure ($T_{Failure}$)	$T_{Failure}$: Failure measure that is opted in case of deciding which priority task should be executed it cannot be decided based on surplus time.
Task scheduling time ($R_{SchedTime}$)	$R_{SchedTime}$: specifies the time at which the particular task is to be scheduled on a physical resource. This value is obtained after execution of the proposed optimal scheduling algorithm.

The following equation is used to compute the task's finish time, where i determines the number of tasks, k represents the number of timeslots taken by the task until its complete execution in this preempt scheduling mode, where *i* is the final timeslot taken by the task. First, the task finish time is computed using equation 1.

$$T_{finish_i} = \sum_{k=1}^{l} (T_{arrival_{ik}} + T_{duration_{ik}}) \qquad (1)$$

The following equation computes the surplus time of tasks by subtracting the worst-case finish time from its duration.

$$T_{spt_i} = \sum_{k=1}^{l} (T_{wcft_{ik}} - T_{duration_{ik}}) \qquad (2)$$

Following is the objective function focuses on minimizing the task surplus time, T_{spt_i}, as shown in equation (3) below:

$$minimize(T_{spt_i}) = \min \left\{ \sum_{k=1}^{l} (T_{wcft_{ik}} - T_{duration_{ik}}) \right\} \qquad (3)$$

As explained above, task surplus time depicts the timeslot for which a particular task can wait. The objective function focuses on minimizing task surplus time, T_{spt_i}. It is done based on the assumption that minimizing the surplus time can play an influential role in deciding the perfect task execution order according to the current situation. For instance, consider a scenario wherein t1 and t2 are two candidate tasks having the surplus time of 4 and 10, respectively. Now, if t1 gets an execution chance and no task arrives within the surplus time of t2, and t2 gets an execution chance after 6, the resource will remain idle during this period. In this regard, minimizing the task surplus time can lead to effective task scheduling having less idle tasks. However, one point must be considered here: optimized surplus time of t2 must not be less than t1. Therefore, we impose the following constraints on the surplus time of two candidate tasks in the waiting queue.

- Among two candidate tasks, min(T_{spt}) of the task having lowest surplus time must ≤ min (T_{spt}) of the task with maximum surplus time.
- $0 < T_{arrival} < T_{duration}$
- $T_{duration} > 0$

The primary purpose of harnessing the surplus measure is to have an optimal decision policy to select among two candidate tasks. However, sometimes a situation arises wherein it becomes indecisive about considering surplus time as a task selector indicator. For instance, consider a scenario when both tasks' optimized surplus value leads to an identical optimized value. In such a scenario, we opt for exigency and failure measures to select a task in the situation mentioned above. The exigency measure determines the tasks between simple event-driven tasks and periodic priority tasks. Failure measure is harnessed to choose the task between simple periodic and starving tasks.

The concept of exigency measure and failure measure is employed by following the assumptions of []. First, the urgency measure is calculated using the equation below:

$$T_{Exigency} = T_{IdleTime} - T_{duration} \qquad (4)$$

Wherein idle time of task is computed using the following equation:

$$T_{IdleTime} = \left| \sum_{k=1}^{l} (T_{arrival_{ik}} - T_{duration_{ik}}) \right| \qquad (5)$$

Similarly, the following algorithm demonstrates the computational flow of the failure measure to decide whether to execute the current priority task or not. The objective of FM is to analyze whether executing a low priority starving task would not harm the execution of high priority periodic task or not.

Moreover, safe execution of the current periodic task is also ensured with the help of slack time. Slack time is computed using the equation below:

$$T_{SlackTime_i} = T_{SlackTime_i} + T_{ExecutionTime_i} \qquad (6)$$

Wherein $T_{ExecutionTime_i}$ is computed using following equation.

$$T_{ExecutionTime_i} = PT_{duration_i} + PT_{ExecutionTime_i} \qquad (7)$$

Furthermore, it determines task slack for each given periodic task to ensure the safe execution of a task. Slack time can be calculated as the difference between task deadline time and task execution time. The decision regarding task execution is taken in the form of binary values, i.e., 0 or 1. For instance, if the slack time of a periodic task results in a value greater than 0, then FM is assigned the value of 1, otherwise 0. A task holds certain parameters, such as Task ID, task name, creation time, arrival time, deadline, execution time, priority, deadline and phase, as shown in the flowchart in Figure 6. In the first step of the proposed scheduling algorithm, tasks are extracted from the ready queue based on their arrival time. If the arrived task is a prompt event-driven task (PEDT), it is executed within the specified time interval without any delay. Next, if there are two candidate tasks form which one is simple event driven task (SET) and another is periodic priority task (PPT) then the algorithm computes the surplus time of both the tasks and forwards to the optimization module wherein the surplus values are minimized using the proposed objective function implemented using particle swarm optimization (PSO). If the minimized surplus time of SET is less than the minimized surplus time of PPT, then SET is executed, if the surplus time of SET is greater than the surplus time of PPT then PPT is executed. However, if both tasks contain identical surplus time values, then the exigency measure is calculated using equation 4. The exigency measure decides which task to select from SET and PPT. If the exigency measure results in 0 then the PPT with the nearest deadline is executed, otherwise SET with the nearest deadline is executed. If two candidate tasks are PPT and starving task (ST), then optimized surplus time of both the tasks are obtained for task selection. Suppose periodic task has less optimal surplus time than ST. In that case, periodic task is executed, if greater, then PT is executed, and if the surplus time of both the tasks is equal, then failure measure is opted to decide the task selection. The failure measure harnesses task history and profiles log information of both the candidate tasks to pick the best task for execution, as explained in [40]. The proposed scheduling mechanism results in an updated order of the scheduled tasks that consume the physical IoT resources in the best possible way. The scheduling process results in two additional parameters, task priority and execution time, utilized by the task mapping module. Finally, the task scheduler manager (TSM) maintains the scheduling tasks in the task mapping repository. The scheduled tasks are saved in a tuple encompassing $<T_{id,}\ VO_{id},\ R_{SchedTime},\ T_{Priority}>$.

Algorithm: Hybrid Optimal Time-Constraint Scheduling Algorithm

```
1:  function OSchedule(Tasks [0....m − 1], $Task_i$ )
2:  Input: Array Tasks [0...n − 1] of the given tasks and $Task_i$ is the time step at which tasks arrived in the queue for the execution
3:      $Failure_{measure} \leftarrow 0$,
        $Urgent_{measure} \leftarrow$ Taskdeadline - $Task_{FinishTime}$,
        $Surplus_{time} \leftarrow wcft - d$
4:      if $Tasks_{nature} = PEDT$ then
5:          Execute PEDT
6:      else if $Tasks_{type} = SET$ then      //simple event-driven task
7:          if (exists (PPT)==true)      //if priority periodic task also exists
              Pick_Task (SET, PPT)
8:            else if min(sptSET) == min(sptPPT)
9:            Compute $Urgent_{measure}$ using Equation
10:               if Urgencymeasure == 0 then
11:               Execute the PPT with nearest deadline
12:              else
13:               Execute the SET with nearest deadline
14:               end if
15:           end if
16:         else
17:             Execute the SET with nearest deadline
18:         end if
19:     else if $Task_{nature} = SPT$ then      //if task is simple periodic task
20:         if (exists (strT)==true)      //if a starving task is waiting in the ready queue
21:             Pick_Task (SPT, sprT)
22:               else if min(sptSPT) == min( sptsprT)
23:                 Calculate $Failure_{measure}$ using Algorithm 1
24:                     if Failuremeasure = = 0 then
25:                             Execute Low Priority Starving Task strT
26:                     else
27:                       Execute the High Priority PT
28:                     end if
29:                 end if
30:             Execute the High Priority PT
31:         end if
32:     else
33:         Invalid Task Type
34:    end if
35: end function
```

The following fucntion "*Pick_Task*" decides the task to be executed from two candidate tasks in the queue. First, it takes two tasks as an input and computes their surplus time by subtracting the worst-case finish time from its duration. Next, surplus time of both the tasks is minimized by implementing the objective function. Firstly, the algorithm

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 10 February 2022 doi:10.20944/preprints202202.0150.v1

compares both tasks' optimized surplus time (i.e., T1 and T2). If the surplus time of T1 is less than the surplus time of T2 then T1 is selected for execution; if the surplus time of T1 is greater than the surplus time of T2 then T2 is opted for execution.

```
        Input: Task 1, Task 2
        Output:  Select one task from two given tasks
          Pick_Task (T1, T2)
1:      S(T1)=compute {spt(T1)}
2:      S(T2)=Compute {spt(T2)}
3:      X  = PSO {S(T1)}
4:      Y  = PSO {S(T2)}

5:            if X < Y then
6:            Execute T1
7:            else if  X > Y then
8:            Execute T2
```

3.3.6 Allocation and Deployment Process of Scheduled Energy Management Tasks

The next step after task scheduling is to allocate the task to physical IoT devices. The tasks scheduled by the optimal schedular module are given as an input to the task allocation manager (TAM), which allocates them to the designated physical IoT devices. All the allocation-related information is stored in TAM, obtained from task repository, virtual object (VO) repository, and scheduling repository to assign tasks to the associated sensors and actuators. For successful task deployment, it is essential to have information about the IoT device status during the allocation phase. Therefore, the task deployer processes the information about device status and the optimal scheduling order to allocate the tasks on physical IoT devices. For instance, the task “getSolar” is deployed to the corresponding physical device, i.e., photovoltaic (PV), using the task scheduling results obtained from the proposed optimal time-constraint-based scheduling algorithm. 1. Experimental details and performance analysis. The proposed optimal task scheduling method is implemented using different tools and technology as illustrated in Table 2. The experiments are conducted on real-time device data of smart home using IMU sensor that captures real time energy data of a smart home. Raspberry-pi and PC server are used as hardware running Microsoft windows 10 operating system on it. The sensors include temperature sensor, wind sensor, humidity sensor, battery sensor and diesel sensor. For implementation, we use python, C#, Javascript, HTML, and CSS programming languages to implement task orchestration executed on PyCharm Professional 2020, Python Flask, and visual studio. Raspberry PI is attached to the personal computer (PC) to acquire the sensing data and store it in MySQL. The IMU sensor is connected to the Raspberry PI to get the real energy data from nano-grid house. The data from the sensor is processed to implement task orchestration having optimal scheduling mechanism. All the employed tools and techniques are illustrated in the following table.

3.3.7 Sensor Readings

The visualization of sensor readings employed to get the data from physical resources such as PV, WT, ESS and diesel are shown in Figure 7. The total capacity of PV and WT is to produce energy up to 1500 kWh. Similarly, ESS and diesel have 4500 and 2500 capacities, as shown in Figure 7. The PV readings are shown with red radial lines, WT readings are shown with orange streaks, ESS readings are shown with green lines, and diesel readings are shown with red stripes. Accordingly, these values are passed to the EMS module to make an operational plan. For instance, state-of-charge (SoC) obtained from ESS determines the physical devices to employ for meeting the load demand and utilizing the maximum amount of energy simultaneously.

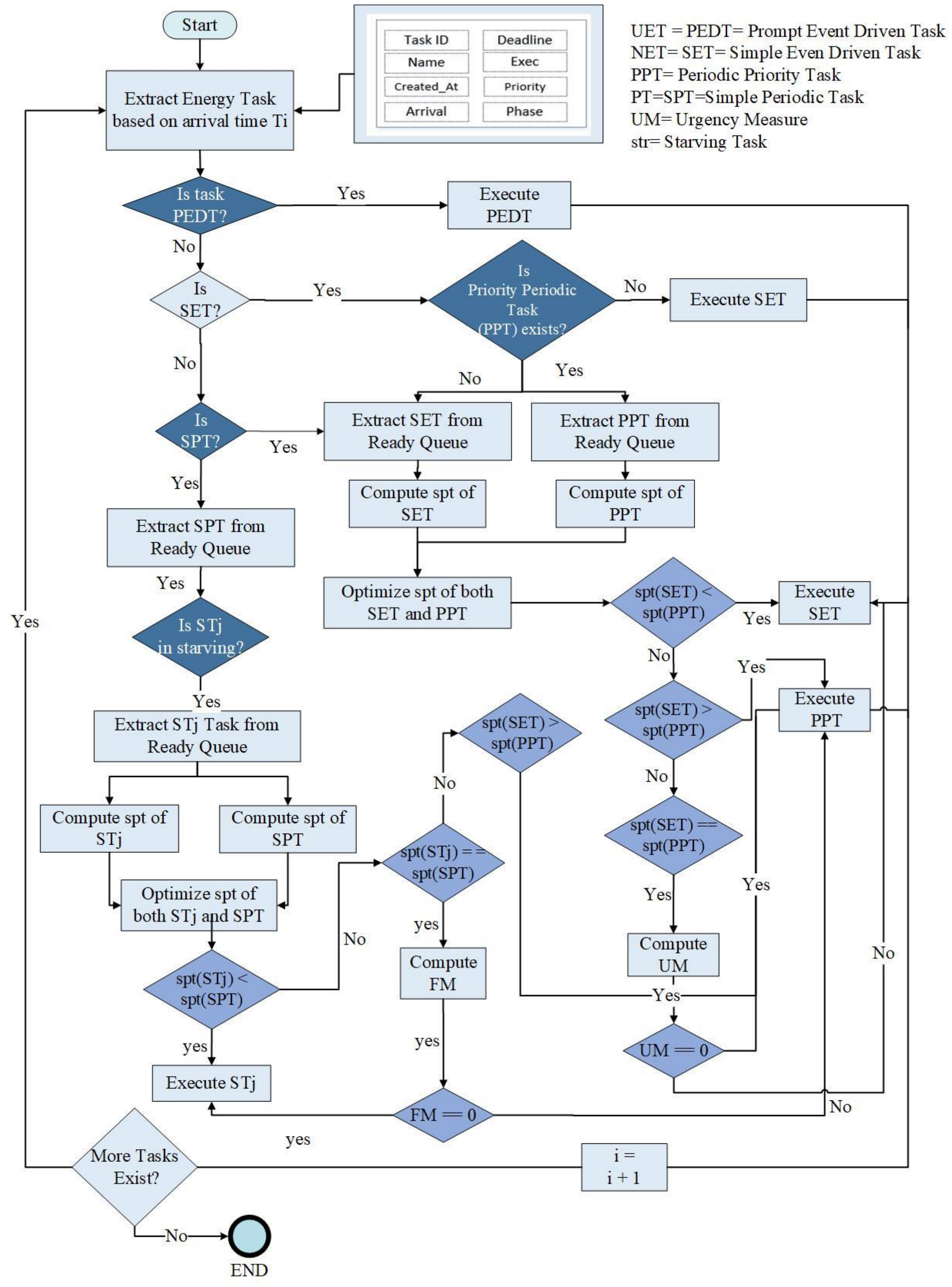


Figure 6 Flowchart of Proposed Scheduling Algorithm in IoT Task Orchestration

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 10 February 2022 doi:10.20944/preprints202202.0150.v1

Table 2 Tools and Techniques

Tool/Technology	Details
Data nature	Real-time device data obtained using IMU sensor
Operating System	Microsoft Windows 10
Hardware	Raspberry-PI, PC
IoT sensors/resources	Temp sensor, wind sensor, humidity sensor, battery sensor, diesel sensor
Programming Tool /Framework	PyCharm Professional 2020, Python Flask, HTML, and CSS
Server	Flask
Programming Language	Python, C#, Javascript, HTML, CSS
Libraries	Drools, JSPlumb, Accord
Core Programming Language	Python, JavaScript and jQuery
Front End Framework	Bootstrap
Database	MySQL

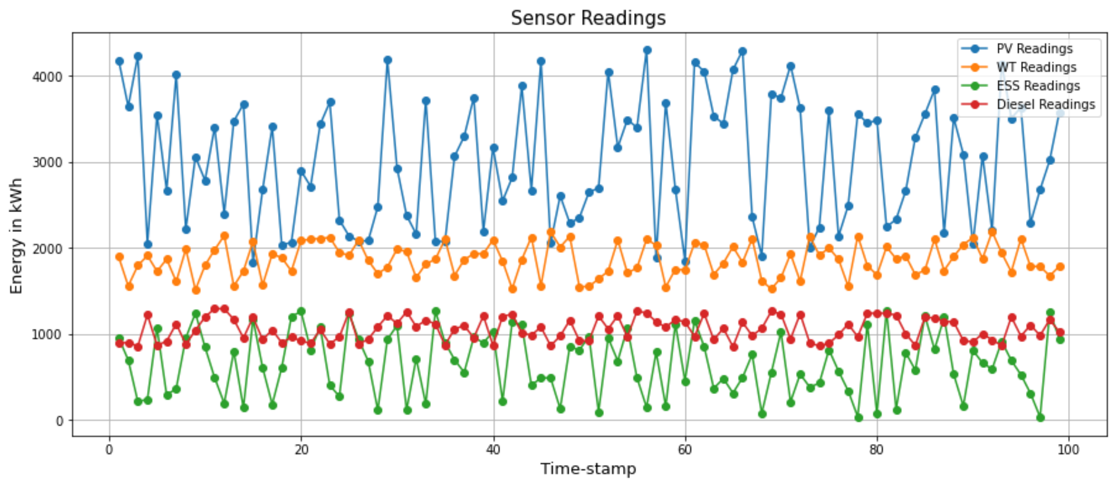
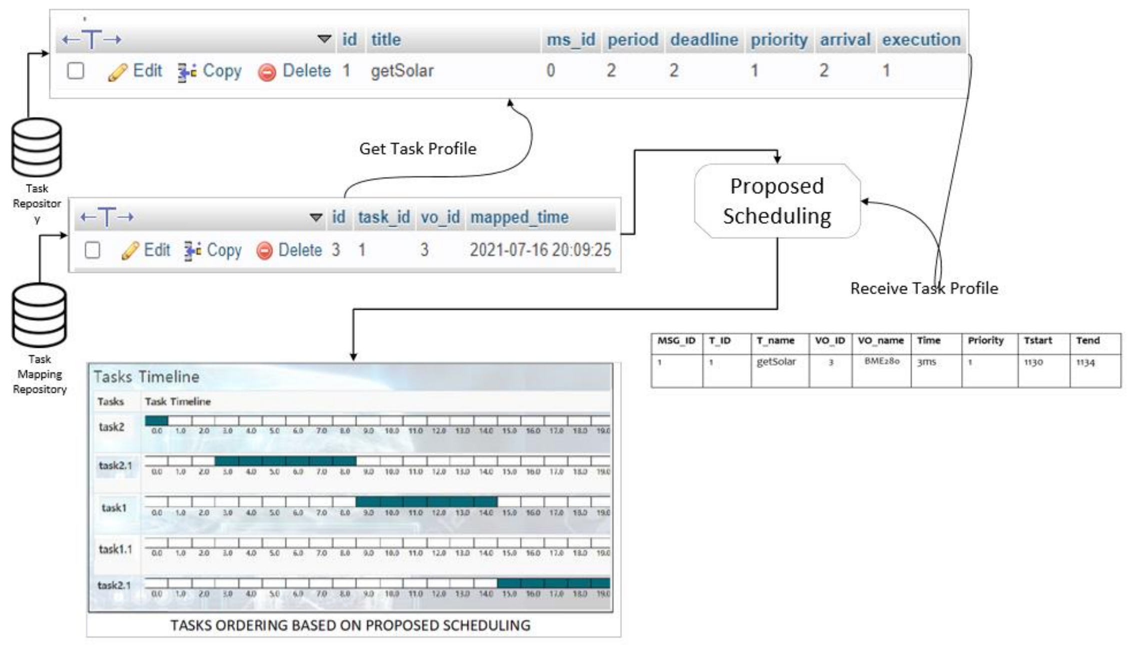


Figure 7 Data Readings from PV, WT, ESS and Diesel

4. Performance Analysis

Before evaluating the results, let’s have a look into the schedulig flow further with the help of figure 8. The mapping information is employed to locate the task and VO profile in the mapping repository. It can be seen in the figure that the task ID 1 is tracked to obtain the task profile of (getSolar). The task has priority of 1 and arrival of 2. The information encompassing a tuple (1, 3, mapped time) is passed to the propsoed scheduling technique. The proposed scheduling technique harnesses this information to determine the task execution order, as shown in the figure 8.



id	title	ms_id	period	deadline	priority	arrival	execution
1	getSolar	0	2	2	1	2	1

id	task_id	vo_id	mapped_time
3	1	3	2021-07-16 20:09:25

MSG_ID	T_ID	T_name	VO_ID	VO_name	Time	Priority	Tstart	Tend
1	1	getSolar	3	BME280	3ms	1	1130	1134

Figure 8 Execution flow of Task scheduling

Following are the details about the proposed optimal scheduling technique regarding response time, latency, throughput, and round-trip time (RTT) analysis. Firstly, let us look into the results obtained by optimizing the surplus time. As explained earlier, this study focuses on minimizing the task surplus time to evade the resources from being idle. Figure 9 exhibits the optimization results of task surplus time. It is evident from the figure that at each level, the cost is reduced since the focus of the proposed objective function is on minimizing the cost of task surplus time. We have considered 50 particles and 20 iterations. Although surplus time's initial value is 2, the PSO directs the particles towards the optimal direction to find the best solution. It is evident that initially, PSO quickly locates the best solution in earlier iteration; however, the performance deteriorates while reaching to the optimal solution. The round trip time analysis (RTT) determines the timespan consumed by the proposed model spanning from commencing, i.e., energy task generation to deploying them to physical IoT devices. The performance is evaluated in minimum, average, and maximum round trip time (RTT) analysis.

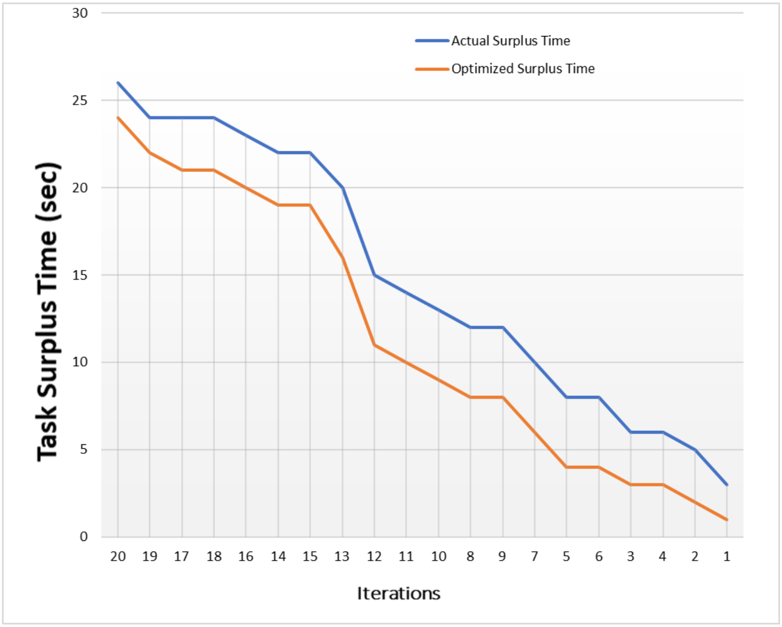


Figure 9 Task Surplus Time Optimization

As explained earlier, contemplating the nature of smart gid based systems, the tasks are divided into event-driven and periodic tasks. For periodic tasks, the minimum reported value of RTT is 6 ms (requestWindData), the average value of RTT is 10 ms (getSolarData) the value of maximum RTT is 18.5ms (requestESSData), as shown in Figure 10.

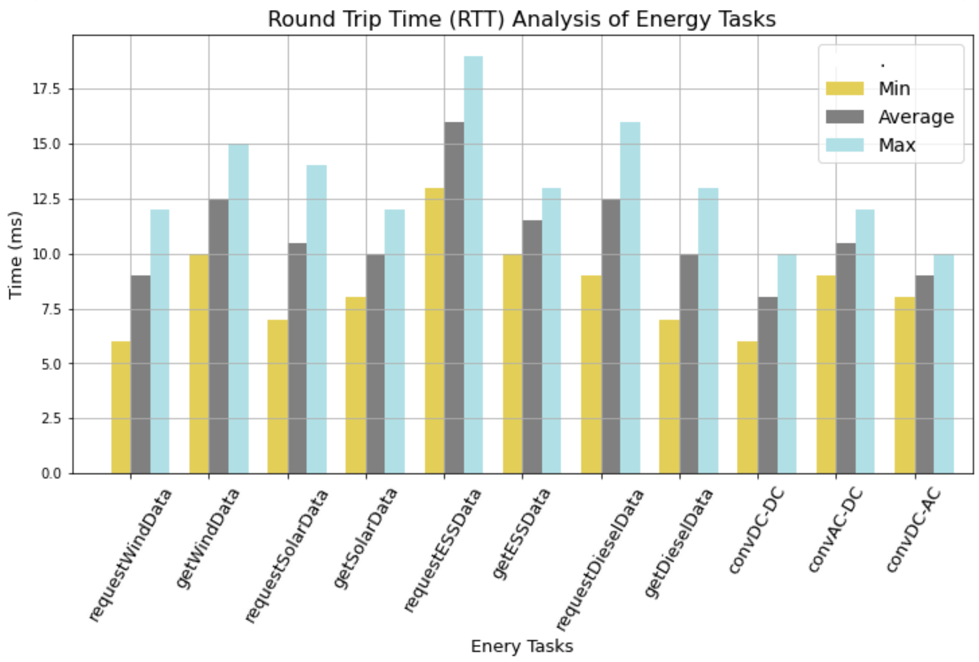


Figure 10 Round Trip Time Analysis of Energy Tasks

The
second evaluation measure is throughput, which is shown in Figure 11. The x-axis shows the number of smart homes, and the y-axis shows the throughput analysis depicting the number of energy tasks executed within one second. Thus, the x-axis represents the throughput values in ms, and the y-axis represents the no. of homes. As explained earlier, a total of 24 houses is considered, and throughput analysis is determined by dividing the number of houses into three groups: (1) 8 houses, (2) 16 houses and, 3) 24-houses as shown in the figure below. The throughput analysis represents minimum, average, and maximum throughput values according to the said categories. The minimum, average, and maximum throughput values lie between 18-28 no. of tasks for 16 houses, 32-57 for 16-houses, and 43 to 67 for 24-houses.

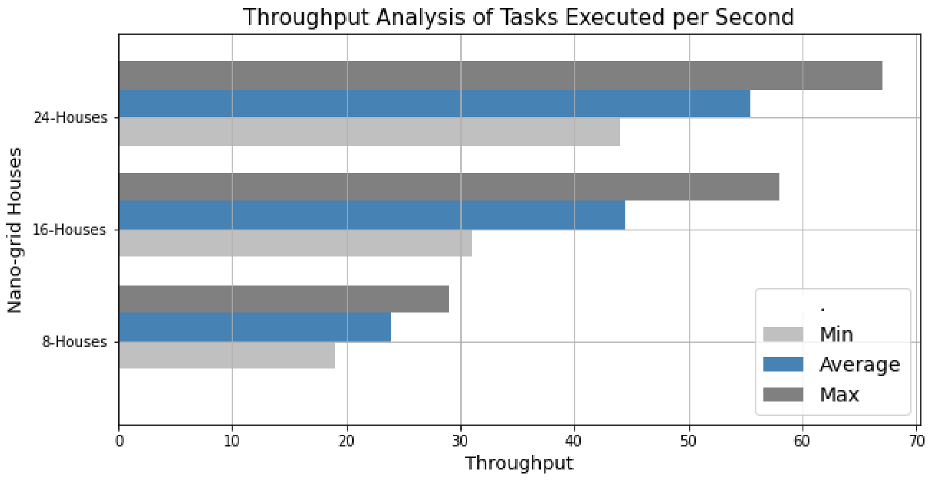


Figure 11 Throughput Analysis of Energy Tasks

It is evident from the figure that the value of throughput increases as the number of smart homes increases from 8 to 24. Figure 12 envisions the results in terms of task latency. Like RTT and throughput, latency is also analyzed in maximum, moderate, and maximum values. We have focused on data of three different smart homes to measure the task latency. The minimum latency value is 4.28 ms, moderate is 15.49 ms, and the maximum is 37.52 ms. The latency value increases with the increase in the number of homes. For instance, the moderate value of average latency reached from 8.4 ms to 17 ms from 8 to 24 houses, respectively. The outcomes of analysis exhibit that the model performs ideally when the number of smart homes increases which validates that the proposed model can be adopted to make the energy management system of a smart home robust and intelligent.

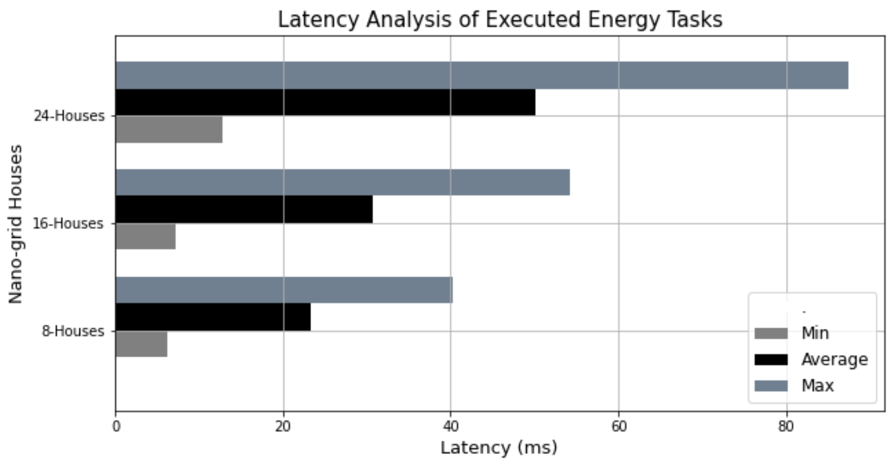


Figure 12 Task Latency of Energy Tasks

Another evaluation measure employed to evaluate the proposed model is response time (RT). The RT helps in specifying the total time consumed by a particular process from start time to its completion time, often referred as elapsed time. We have drawn RT analysis here in terms of tasks categories, i.e., event-driven and periodic tasks. This analysis is presented to evaluate the executed energy tasks in terms of average response time. Event-driven tasks' average response time values fluctuate between 20 to 79 ms. The average response time for periodic tasks varies from 50 to 140 ms, as shown in Figure 13. It is evident that the average response time of event-driven tasks is lower than periodic energy tasks as the periodic tasks have high priority as compared to sensing energy tasks.

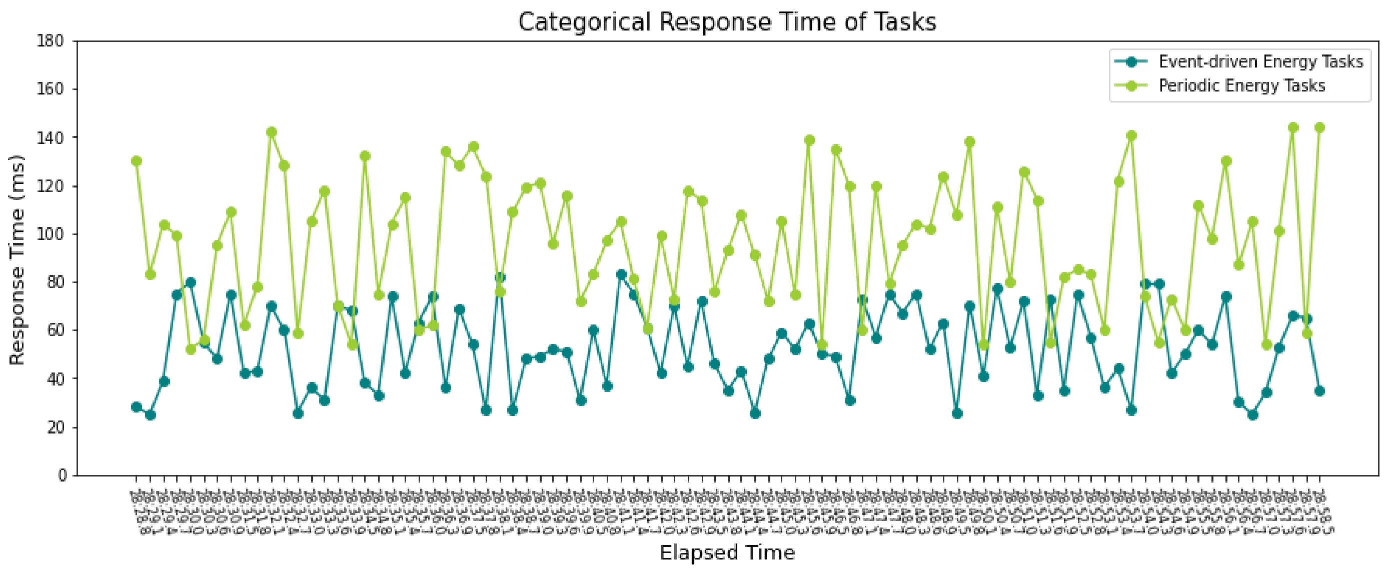
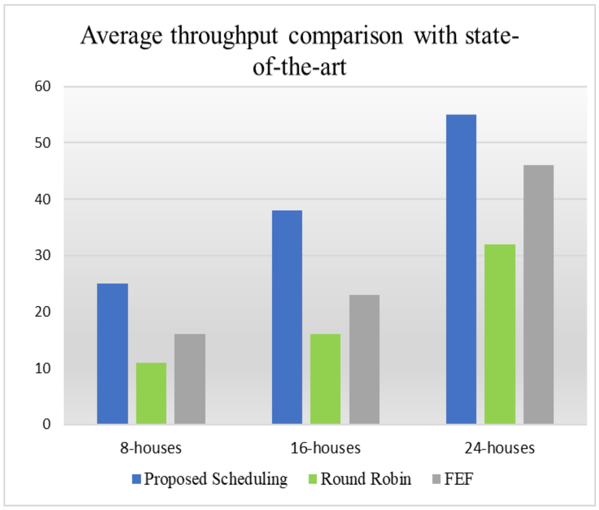


Figure 13 Categorical Analysis of Energy Tasks in terms of Response Time

To better demonstrate the effectiveness of the proposed model, we have drawn a comparison of the proposed optimal scheduling algorithm with the contemporary state-of-the-art algorithms, such as round-robin (RR) and fair emergency first (FEF), as shown in Figure 14(a&b) and 15. The x-axis shown in Figure 14(a) shows number of houses, and y-axis shows no. of tasks. In Figure 14(b), x-axis shows the number of houses and y-axis shows time in miliseconds. Average throughput comparison is performed for three categories of houses (i.e., 8, 16 and 24), as shown in Figure

14(a). It can be seen that the proposed model executed higher number of tasks than other two models, i.e., 25 vs 11 and 16 for 8 houses, 38 vs 16 and 23 for 16 houses, and 55 vs 32 and 46 for 24-houses. Similarly, average latency (delay) analysis shown in Figure 14(b) shows that the propsoed model consumed least amount of time than other two scheduling algorithms, such as 8.4 ms vs 18.7 and 19.3 ms, 15.49 vs 36.4 and 22.8, and 17.3 vs 44.1 and 26.3 for 8, 16 and 24 houses, respectively.



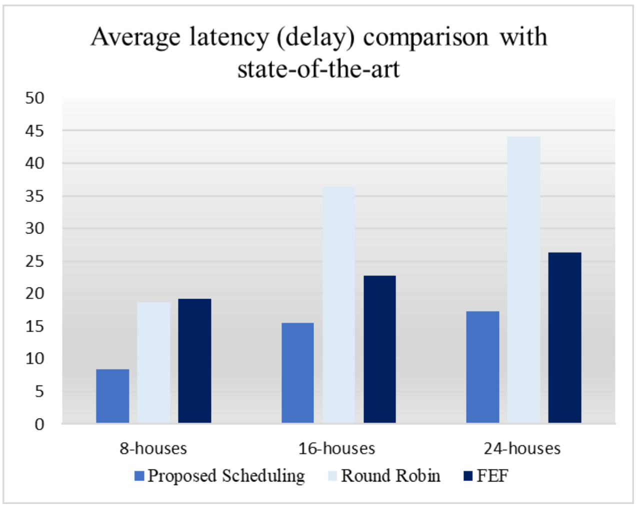


Figure 14(a) Throughput comparison with state-of-the-art scheduling algorithms, (b) Latency comparison with state-of-the-art scheduling algorithms

Afterward, the comparison is drawn in terms of task drop and starvation rates, as shown in Figure 15. The proposed optimal scheduling model's task starvation and dropout/failure rates are 12% and 19%, respectively. Conversely, the task starvation produced by RR is 28%, and the task failure rate is 37%. FEF results in 34% task failure rate and 24% task starvation rate. Thus, the proposed model significantly reduces the task starving and failure rate, which enhances the performance of the proposed smart grid-based scheduling mechanism.

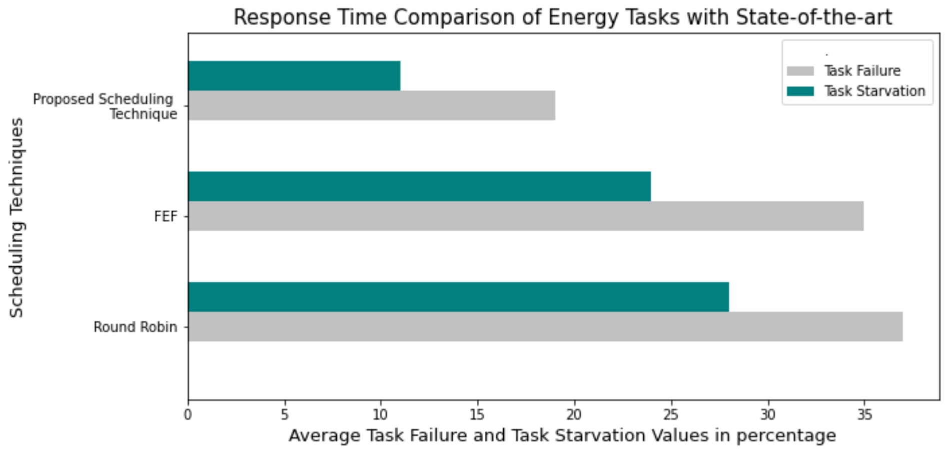


Figure 15 Comparison with existing state-of-the-art in terms of Response Time

Figure 16 demonstrates the results of sensors fault recovery wherein the x-axis represents the simulated number of sensing devices employed for nano-grid energy management. The y-axis represents the time consumed in recovering the sensor faults. Finally, the y-axis represents the actual and optimized recovery time in which a faulty sensor is replaced with a backup sensor to monitor a patient's data effectively. The average recovery time is 32.23 minutes.

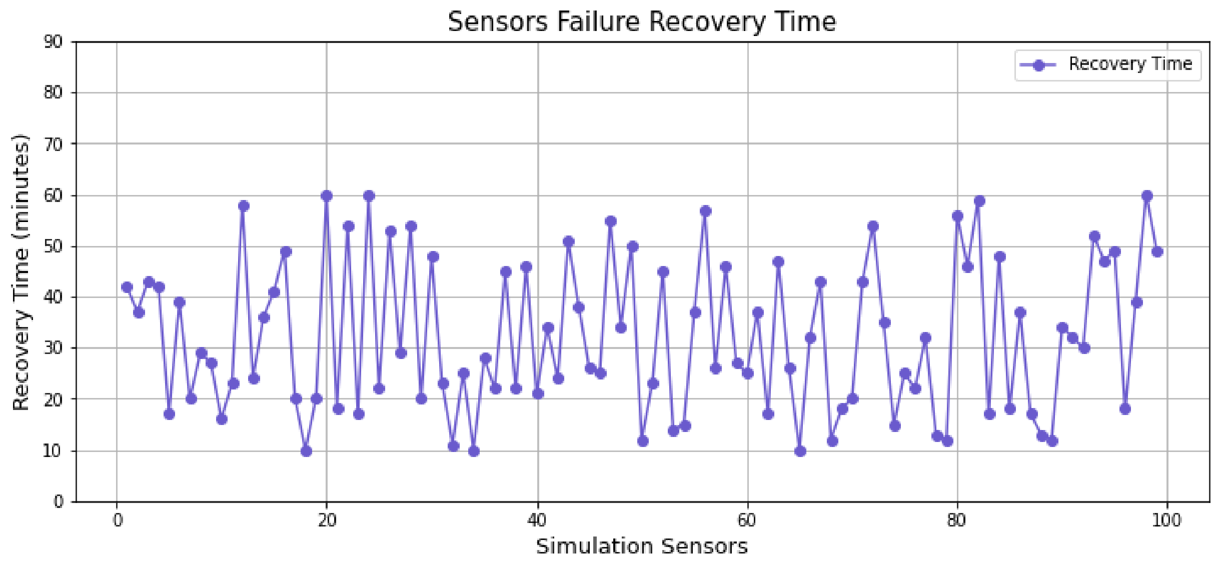


Figure 16 Sensor Fault Recovery Analysis

5. Conclusion

This paper has presented IoT task orchestration based nanogrid energy management systems and optimal time-aware scheduling mechanism designed to evade the issues of missing task deadlines in real-time mission-critical IoT applications. The system implements an IoT-enabled task orchestration framework that automatically generates energy tasks by analyzing the service information provided by a user, schedules energy tasks by considering the time constraints, and dynamically deploys them to physical resources based on highly prioritized energy tasks. The optimization module of the proposed system comprises two core optimization methods wherein an objective function is designed to address two optimization problems: (1) energy management of a smart grid-based framework to maximize renewable energy resources and minimize the use of non-renewable energy resources to provide a cost-effective and eco-friendly solution, and (2) optimizing the surplus time of energy tasks so that starving tasks also get a fair chance to be executed. The objective function of both the modules is implemented using the particle swarm optimization (PSO) algorithm. Moreover, the proposed model stipulates an NLP-based autonomous task mapping system that automatically map the generated energy tasks to the corresponding virtual objects. The model commences from implementing an energy management framework for a nano-grid house and incorporating the relevant tasks in the list of generated tasks, virtual objects creation by replicating the physical resources, autonomous mapping of tasks to corresponding virtual objects, scheduling the tasks to determine best execution order of the tasks, allocation of the tasks to corresponding sensors and their dynamic deployment to the physical resources. The standard performance analysis measures including RTT, throughput, latency, response time, and task starvation rate are employed to assess the results. The study's outcomes suggest that the proposed optimal scheduling significantly decreases the task failure and starvation rate compared to contemporary task scheduling methods such as FEF and round-robin. The proposed optimal scheduling model's task dropout/failure and starvation rate are 12% and 19%, respectively which yields that the model reduces the task starvation rate by (16%) compared to RR and (12%) compared to FEF scheduling. These results reveal that the proposed optimal scheduling method provides a sustainable and effective solution to real-time based smart grid industry in effectively handling the allocation process of the emergent energy tasks dynamically.

References:

[1] Imran; Ghaffar, Z.; Alshahrani, A.; Fayaz, M.; Alghamdi, A.M.; Gwak, J. A Topical Review on Machine Learning, Software Defined Networking, Internet of Things Applications: Research Limitations and Challenges. Electronics **2021**, 10, 880.

[2] Sicari, S.; Rizzardi, A.; Grieco, L.A.; Coen-Porisini, A. Security, privacy and trust in Internet of Things: The road ahead. Comput. Netw. **2015**, 76, 146–164.

[3] Zanella, A.; Bui, N.; Castellani, A.; Vangelista, L.; Zorzi, M. Internet of things for smart cities. IEEE Internet Things J. **2014**, 1, 22–32.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 10 February 2022 doi:10.20944/preprints202202.0150.v1

[4] Iqbal, N., Ahmad, S., Ahmad, R., & Kim, D. H. (2021). A Scheduling Mechanism Based on Optimization Using IoT-Tasks Orchestration for Efficient Patient Health Monitoring. *Sensors*, *21*(16), 5430.

[5] Ahmad, S., Khudoyberdiev, A., & Kim, D. (2019). Towards the task-level optimal orchestration mechanism in multi-device multi-task architecture for mission-critical IoT applications. IEEE Access, 7, 140922-140935

[6] Ahmad, S., & Kim, D. (2020). A multi-device multi-tasks management and orchestration architecture for the design of enterprise IoT applications. Future Generation Computer Systems, 106, 482-500.

[7] S. Ahmad, F. Mehmood, and D.-H. Kim, "A DIY approach for the design of mission-planning architecture using autonomous taskobject mapping and the deployment model in mission-critical IoT systems," Sustainability, vol. 11, no. 13, p. 3647, Jul. 2019.

[8] Zheng, J.; Simplot-Ryl, D.; Bisdikian, C.; Mouftah, H.T. The internet of things [Guest Editorial]. IEEE Commun. Mag. **2011**, 49, 30–31.

[9] O. Vermesan, P. Friess, P. Guillemin, S. Gusmeroli, H. Sundmaeker, A. Bassi, I. S. Jubert, M. Mazura, M. Harrison, M. Eisenhauer, and P. Doody, ``Internet of Things strategic research roadmap," in Internet of Things: Global Technological and Societal Trends, vol. 1, O. Vermesan, P. Friess, P. Guillemin, S. Gusmeroli, H. Sundmaeker, and A. Bassi, Eds. Gistrup, Denmark: River, 2011, pp. 952.

[10] P. Friess, Internet Things-Global Technological Societal Trends From Smart Environments Spaces to Green ICT. Gistrup, Denmark River, 2011.

[11] F. Xia, L. T. Yang, L. Wang, and A. Vinel, ``Internet of things," Int. J. Commun. Syst., vol. 25, no. 9, p. 1101, Sep. 2012

[12] Öhrvik, H., Aaseth, J., & Horn, N. (2017). Orchestration of dynamic copper navigation–new and missing pieces. *Metallomics*, *9*(9), 1204-1229

[13] Iqbal, N., & Kim, D. H. (2021). IoT Task Management Mechanism Based on Predictive Optimization for Efficient Energy Consumption in Smart Residential Buildings. *Energy and Buildings*, 111762.

[14] IEA. World Energy Outlook 2020 -Summary. Report 2020:1–25.

[15] Ahmad, S., Ullah, I., Jamil, F., & Kim, D. (2020). Toward the Optimal Operation of Hybrid Renewable Energy Resources in Microgrids. Energies, 13(20), 5482.

[16] Lasseter, R. H. (2002, January). Microgrids. In *2002 IEEE Power Engineering Society Winter Meeting. Conference Proceedings (Cat. No. 02CH37309)* (Vol. 1, pp. 305-308). IEEE.
[17] A. Werth, N. Kitamura, and K. Tanaka, "Conceptual study for open energy systems: Distributed energy network using interconnected DC nanogrids," IEEE Trans. Smart Grid, vol. 6, no. 4, pp. 1621–1630, Jul. 2015.

[18] Qayyum, F., Jamil, F., Ahmad, S., & Kim, D. H. (2022). Hybrid Renewable Energy Resources Management for Optimal Energy Operation in Nano-Grid.

[19] Shi, W.; Xie, X.; Chu, C.-C.; Gadh, R. Distributed optimal energy management in microgrids. IEEE Trans. Smart Grid 2014, 6, 1137–1146.

[20] García-Vera, Y.E.; Dufo-López, R.; Bernal-Agustín, J.L. Optimization of isolated hybrid microgrids with renewable energy based on different battery models and technologies. Energies 2020, 13, 581.

[21] Wang, X.; Palazoglu, A.; El-Farra, N.H. Operational optimization and demand response of hybrid renewable energy systems. Appl. Energy 2015, 143, 324–335.

[22] Stavrinides, G.L., Karatza, H.D. Orchestrating real-time IoT workflows in a fog computing environment utilizing partial computations with end-to-end error propagation. *Cluster Comput* **24,** 3629–3650 (2021). https://doi.org/10.1007/s10586-021-03327-y

[23] K. Velasquez, D. P. Abreu, D. Gonçalves, L. Bittencourt, M. Curado, E. Monteiro, and E. Madeira, ``Service orchestration in fog environments," in Proc. IEEE 5th Int. Conf. Future Internet Things Cloud (FiCloud), Aug. 2017, pp. 329336.

[24] Z. Ding, K. Ota, Y. Liu, N. Zhang, M. Zhao, H. Song, A. Liu, and C. Zhiping, ``Orchestrating data as a servicesbased computing and communication model for information-centric Internet of Things," IEEE Access, vol. 6, pp. 3890038920, 2018.

[25] Yao, S., Hao, Y., Zhao, Y., Shao, H., Liu, D., Liu, S., ... & Abdelzaher, T. (2020, August). Scheduling real-time deep learning services as imprecise computations. In *2020 IEEE 26th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)* (pp. 1-10). IEEE.

[26] How far is the Hype of IoT. Accessed Mar. 15, 2019. [Online]. Available: https://www.rcrwireless.com/20160628/opinion/reality-check- 50b-iot-devices-connected-2020-beyond-hypereality-tag10

[27] F. Xia, L.T. Yang, L. Wang, A. Vinel, Internet of things, Int. J. Commun. Syst. 25 (2012) 1101.

[28] O. Vermesan, P. Friess, P. Guillemin, S. Gusmeroli, H. Sundmaeker, A. Bassi, I.S. Jubert, M. Mazura, M. Harrison, M. Eisenhauer, et al., Internet of things strategic research roadmap, Internet Things-Global Technol. Soc. Trends 1 (2011) 9–52.

[29] N. Zhang, P. Yang, J. Ren, D. Chen, L. Yu, X. Shen, Synergy of big data and 5g wireless networks: opportunities, approaches, and challenges, IEEE Wirel. Commun. 25 (2018) 12–18

[30] P. Vlacheas, R. Giaffreda, V. Stavroulaki, D. Kelaidonis, V. Foteinos, G. Poulios, P. Demestichas, A. Somov, A.R. Biswas, K. Moessner, Enabling smart cities through a cognitive management framework for the internet of things, IEEE Commun. Mag. 51 (2013) 102–111.

[31] F.S. Aliee, H. Kashfi, B. Farahani, The evolving enterprise architecture: A digital transformation perspective, in: Proceedings of the International Conference on Omni-Layer Intelligent Systems, ACM, 2019, pp. 179–183.

[32] Enterprise internet of things (enterprise IoT), 2018, http://www.enterox.com/IoT/articles/enterprise-internet-ofthings. htm, [Online; accessed 29-August-2019].

[33] K. Velasquez, D.P. Abreu, D. Gonçalves, L. Bittencourt, M. Curado, E. Monteiro, E. Madeira, Service orchestration in fog environments, in: 2017 IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud), IEEE, 2017, pp. 329–336.

[34] K. Velasquez, D.P. Abreu, M.R. Assis, C. Senna, D.F. Aranha, L.F. Bittencourt, N. Laranjeiro, M. Curado, M. Vieira, E. Monteiro, et al., Fog orchestration for the internet of everything: state-of-the-art and research challenges, J. Internet Serv. Appl. 9 (2018) 14.

[35] X. Liu, Y. Liu, H. Song, A. Liu, Big data orchestration as a service network, IEEE Commun. Mag. 55 (2017) 94–101.

[36] M. Huang, Y. Liu, N. Zhang, N.N. Xiong, A. Liu, Z. Zeng, H. Song, A services routing based caching scheme for cloud assisted CRNs, IEEE Access 6 (2018) 15787–15805.

[37] V. Pilloni, E. Abd-Elrahman, M. Hadji, L. Atzori, H. Afifi, IoT_ProSe: Exploiting 3GPP services for task allocation in the Internet of Things, Ad Hoc-Netw. 66 (2017) 26–39.

[38] Elsts, A.; Bijarbooneh, F.H.; Jacobsson, M.; Sagonas, K. ProFuN TG: A tool for programming and managing 675 performance-aware sensor network applications. 2015 IEEE 40th Local Computer Networks Conference 676 Workshops (LCN Workshops). IEEE, 2015, pp. 751–759

[39] F. Daniel, J. Eriksson, N. Finne, H. Fuchs, A. Gaglione, S. Karnouskos, P.M. Montero, L. Mottola, F.J. Oppermann, G.P. Picco, et al., Make-Sense: Real-World Business Processes Through Wireless Sensor

Networks,CONET/UBICITEC, 2013, pp. 58–72.

[40] Ahmad, S., Ali, J., Jamil, F., Whangbo, T. K., & Kim, D. (2021). Complex Problems Solution as a Service Based on Predictive Optimization and Tasks Orchestration in Smart Cities.

[41] Ahmad, S., & Kim, D. H. (2021). A task orchestration approach for efficient mountain fire detection based on microservice and predictive analysis In IoT environment. *Journal of Intelligent & Fuzzy Systems*, (Preprint), 1-16.

[42] Pandya, B.; Pourabdollah, A.; Lotfi, A. Fuzzy logic web services for real-time fall detection using wearable accelerometer and gyroscope sensors. In Proceedings of the 13th ACM International Conference on PErvasive Technologies Related to Assistive Environments, Corfu Greece, 30 Julne–3 July 2020; pp. 1–7.

[43] Bet, P.; Castro, P.C.; Ponti, M.A. Fall detection and fall risk assessment in older person using wearable sensors: A systematic review. Int. J. Med. Inform. **2019**, 130, 103946.

[44] Saidi, H.; Labraoui, N.; Ari, A.A.A.; Bouida, D. Remote health monitoring system of elderly based on Fog to Cloud (F2C)
computing. In Proceedings of the 2020 International Conference on Intelligent Systems and Computer Vision (ISCV), Fez, Morocco, 2–4 April 2020; pp. 1–7.

[45] S. Ahmad, S. Malik, I. Ullah, M. Fayaz, D.H. Park, K. Kim, D. Kim, An adaptive approach based on resource awareness towards power-efficient real-time periodic task modeling on embedded iot devices, Processes 6 (2018) 90.

[46] Y. Sahni, J. Cao, L. Yang, Data-aware task allocation for achieving low latency in collaborative edge computing, IEEE Internet Things J. 6 (2018) 3512–3524.

[47] W. Li, F.C. Delicato, P.F. Pires, Y.C. Lee, A.Y. Zomaya, C. Miceli, L. Pirmez, Efficient allocation of resources in multiple heterogeneous wireless sensor networks, J. Parallel Distrib. Comput. 74 (2014) 1775–1788.

[48] E.A. Khalil, S. Ozdemir, S. Tosun, Evolutionary task allocation in Internet of Things-based application domains, Future Gener. Comput. Syst. 86 (2018) 121–133.

[49] H. Ali, X. Zhai, U.U. Tariq, L. Liu, Energy efficient heuristic algorithm for task mapping on shared-memory heterogeneous MPSoCs, in: 2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), IEEE, 2018, pp. 1099–1104.

[50] Alsaffar, A.A.; Pham, H.P.; Hong, C.S.; Huh, E.N.; Aazam, M. An architecture of iot service delegation and resource allocation
based on collaboration between fog and cloud computing. Mob. Inf. Syst. **2016**, 2016.

[51] Arabnejad, H.; Barbosa, J.G. List scheduling algorithm for heterogeneous systems by an optimistic cost table. IEEE Trans. Parallel Distrib. Syst. **2013**, 25, 682–694.

[52] Jiang, H.J.; Huang, K.C.; Chang, H.Y.; Gu, D.S.; Shih, P.J. Scheduling concurrent workflows in HPC cloud through exploiting schedule gaps. In International Conference on Algorithms and Architectures for Parallel Processing; Springer: Berlin, Germany, 2011; pp. 282–293.

[53] Agarwal, S.; Yadav, S.; Yadav, A.K. An efficient architecture and algorithm for resource provisioning in fog computing. Int. J. Inf. Eng. Electron. Bus. **2016**, 8, 48.

[54] Pham, X.Q.; Man, N.D.; Tri, N.D.T.; Thai, N.Q.; Huh, E.N. A cost-and performance-effective approach for task scheduling based on collaboration between cloud and fog computing. Int. J. Distrib. Sens. Netw. **2017**, 13, 1550147717742073.

[55] Liu, L.; Qi, D.; Zhou, N.; Wu, Y. A task scheduling algorithm based on classification mining in fog computing environment. Wirel. Commun. Mob. Comput. **2018**, 2018.