Computing and Informatics, Vol. 41, 2022, 1425–1445, doi: 10.31577/cai_2022_6_1425

LOAD BALANCING FOR RESOURCE OPTIMIZATION IN INTERNET OF THINGS (IOT) SYSTEMS

Dorcas Dachollom Datiri, Maozhen Li

Brunel University
Kingston Lane, Uxbridge, London, UB8 3PH, United Kingdom
e-mail: dorcas.datiri@brunel.ac.uk, maozhen.li@brunel.ac.uk

Abstract. Internet of Things (IoT) has been recognised as a promising area for automating numerous processes, however, the major problem with IoT is its potential for rising complexities. Several approaches have moved attention to the edge nodes associated with IoT, hence concepts of edge-computing, resource allocation and load balancing are tantamount to a more robust heterogeneous IoT. The resource optimization terrain comes with several complications for the resource allocation and scheduling algorithms. Load balancing, one of the key strategies for improving system performance and resource utilization in distributed and parallel computing, generally views an effective load balancer as a 'traffic controller' of resources by directing tasks to available and capable resources. In this paper, a framework appropriate for modelling and reasoning about IoT resource optimization is developed. Further, implementation of an optimized resource allocation algorithm taking into consideration the users' quality of experience (QoE) and the quality of service (QoS) is made available. Simulation results authenticate analysis and validate the improved performance over existing work.

Keywords: Load balancing, resource allocation, resource optimization, Internet of Things (IoT), edge computing, scalability

Mathematics Subject Classification 2010: 68Wxx

1 INTRODUCTION

Society today has a great desire for instant gratification, be it in the form of online shopping, booking a table for dinner, bank management or even planning a journey.

As long as these activities require an online presence there is need for use of one or more intelligent device. The use of such devices is synonymous to ample data movement, sharing, manipulation and end-to-end communication between nodes. Although the boom in big data has produced satisfactory data privacy, users are still unintentionally exposed repeatedly to disappointing services and security threats as data is manipulated, owing to excess overhead cost, bottleneck and latency and scalability issues.

Internet of Things (IoT), a fast-evolving concept, has been maturing at a continuous rate, with an increasing number of analyses over the last few years. These analyses have led to several concepts depicting different approaches in making IoT more manageable. A paramount approach is making the edge nodes more efficient by conceptualising and implementing resource allocation. The works of [1, 2, 3, 4, 5, 6, 7, 8] affirm that recent paradigms are shifting concentration from the core structure of IoT to the individual nodes building up the IoT. The approach to tackle issues at the nodal level increasingly shows that the previous thought insurmountable challenges can be broken down and dealt with from the ground level up, using dogmas of divide and conquer algorithms. [2, 8, 9, 10] have put forward the ideology of using edge computing and/or blockchain, they have also compared results to existing works suggesting that their theories and models produce more meaningful outputs. The work of [1] suggests that an effective but challenging way to manage services on edge servers is by keeping the services running effectively via the means of appropriate resources allocation.

In a generation where the use of the internet has become as essential as breathing, a central topology approximately equal to IoT cannot be condoned and so, it is paramount to decentralise functionalities by developing decentralised mechanism and/or topologies that will bring about more effective systems and consequently eliminate major concerns that arise when a server is down.

Figure 1 illustrates how devices, referred to as edge nodes, connect randomly to edge or cloud servers. The random connections equate multi connections and with an addition of just a device to the network comes an exponential increase in number of connections. These connections are multiple and may lead to bottleneck, excess overhead costs, redundancy issues, as well as user dissatisfaction. IoT's increasing popularity poses several challenges, these challenges include increasing demand for higher quality of experience (QoE) and quality of service (QoS), such as high data rates, low communication latency, and low energy cost on data communication and processing [11]. QoS' parameter can be divided into the following forms: RAM (Random Access Memory) parameters; Network bandwidth; Cost; and Completion time [12]. Current works show that QoE's verified management models with subjective tests, do not capture or understand the implications of quality degradation caused by intelligent machines [13]. The explosive growth in IoT necessitates that IoT applications have the capacity to support all connected devices and users without dilapidation in QoE and QoS. Consequently, attempts have been made to develop a scalable, low latency, and optimized resource allocation IoT. Generally, effective load balancers act as 'traffic controllers' of resources. This act directs tasks

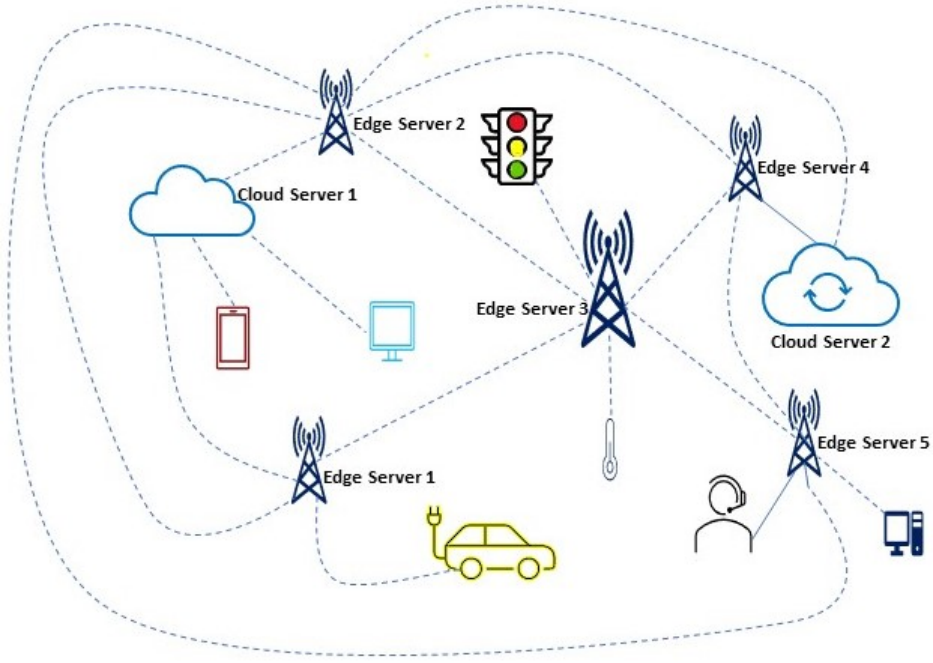

Figure 1. Scenario of existing systems

to available and capable resources, ensuring optimized processing without exertion of excess strain on resources, and resulting in better QoS and QoE. Current research focuses on resource allocation at the detriment of resource scheduling, QoS and QoE. Considering these lapses, a proposed framework of deliberation is one that promotes: a reduction in the number of connections edge nodes establish; decentralizing the IoT's topology; and making IoT scalable. IoT's centralised topology comes with strict hardware specifications that may lead to several inefficiencies. Resource optimization needs to consider the components of the IoT architecture to guarantee efficiency. This paper therefore seeks to address the problems of scalability and latency with intentions to improve users' experiences, and optimization of functionalities to form a solid foundation for an optimal and more secure IoT by decentralising IoT's topology and facilitating geographically challenged machine communication. To answer the major question: what computational and algorithmic theories are suitable, in practice, for management and resource optimisation of Internet of Things? this paper sets to produce the following new contributions:

1. A framework appropriate for modelling and reasoning about IoT resource optimization – improving IoT's current state of the art by producing a decentralised topology ensuring scalability and improved latency;

2. Implementation of an optimized resource allocation algorithm, taking into consideration the users' QoE and the QoS;
3. Experiments and evaluation of developed model, comparison with other approaches and suggestion of further steps to take.

The rest of the paper is organised as follows. Section 2 features related works and motivation. Section 3 introduces the system model. Section 4 presents the implementation of the proposed model, emphasising the architecture with relation to the implementation process. Section 5 presents the experimental results, giving in-depth evaluation. Finally, Section 6 closes the paper.

2 RELATED WORK AND MOTIVATION

This section contains a detailed description of related work. It additionally expatiates the motivation behind the implementation.

2.1 Related Work

Edge computing, a thriving approach to solving the issues of scalability and latency, consists of different paradigms, such as cloudlets, mobile-edge computing, and fog computing; it has several surveys that summarise the architecture of such paradigms [7]. Edge computing uses the Service Level Agreement (SLA) as a commitment between a service provider and a client [1, 2]. Quality of experience (QoE) which is a measure of users' acceptance of applications or services, usually influenced by expectations as well as users' context, is a challenging criterion to gauge as there are no standardized step-by-step platforms for measuring users' quality perceptions, i.e., their quality of experience (QoE), for IoT-based services [14]. Given previous works and set frameworks, the frameworks produced by [5, 14] show some link between service effectiveness and service allocation, these frameworks will govern the measure of QoE in this paper.

IoT itself can simply be viewed as clusters of networks, specifically intranets that are interconnected. The heterogeneity of IoT earns a "significant challenge for the edge cloud to effectively allocate multidimensional limited resources (CPU, memory, storage, bandwidth, etc.) with constraints of applications' quality of service (QoS) requirements" [5]. This heterogeneous nature has incurred a classification of its resources by researchers, a classification of focus is the classification by [2] where "one type of resource corresponds to nodes/things, including the computational resources, storage capacity, and energy resources" and the other type "is the resources corresponding to the communication channel or the network resources, including channel bandwidth, load balancer, and traffic analyser". [5] developed a novel framework named DeepEdge that allocates resources to the heterogeneous IoT applications with the goal of maximizing users' quality of experience (QoE). They achieve this by developing a QoE model that considers aligning the heterogeneous requirements

of IoT applications to the available edge resources and a Q-value (Quality-value) approximation approach to tackle the large space problem of Edge-IoT.

Q-value approximation approach provides a means to control the positive false discovery rate, as Q-values are an estimation of the positive of taking a course of action at point X. The Q factor, a dimensionless parameter, will be the backbone for the proposed generic framework for measuring the QoE of IoT-based services, as depicted in the work of [14], the parameter was further broken into five major steps, specifically: Defining the IoT services and formulating the QoE; Defining the users who implement IoT services; Mean Opinion Score (MOS) survey to indicate the level of the Absolute Category Rating with Hidden Reference (ACR-HR) score; Calculating Differential MOS (DMOS) as the ACR-HR value scale; and Developing strategic implications. Li and Xu in [2] suggest that wireless sensor networks (WSNs) have recently emerged as a platform for several applications, they further suggest that a major problem with WSNs is limited bandwidth. Resource optimization, a term used more frequently in recent research concerning IoT has multiple approaches. These approaches inadvertently have effect on the enterprise architecture, most especially architectures for service-based IoT systems [2]. The resource optimization terrain comes with various complications for the resource allocation and scheduling algorithms. Load balancing, one of the key strategies for improving system performance and resource utilization in distributed and parallel computing, can be divided into two categories: Static Load Balancing (SLB) and Dynamic Load Balancing (DLB). SLB is achieved when the load can be determined and divided by a certain method before execution, whereas DLB is achieved if the system load requires monitoring, and dynamically adjusting while executing [15].

2.2 Motivation and Lapses

The Internet over the years has experienced massive evolutionary trends that have associated the cloud with devices – intelligent devices. This evolution makes cross cultural communication and instant indulgence a norm; these and the ability to control activities directly and/or indirectly as well as automation of daily routines with the aid of intelligent devices connected to actuators and sensors have inadvertently made the world an amazingly comfortable tiny global village. This current invention's bonus to our society does however have its lapses – security, excess computational overhead, scalability, latency and many others. "IoT devices are known to be constraint devices in terms of power, cost, and size. With constraints in place, maintaining security is a challenge" [16]. "Traditional security solutions and protocols cannot be implemented well in IoT specific environment that is typically constrained by limited computing and power resources" [17], furthermore, the heterogeneous nature of IoT devices as well as resources, various perception-action and widely distributed devices and computing resources are other factors that influence IoT based systems. The heterogenous and widely distributed nature of IoT necessitates a topology that accommodates its attributes to ensure an optimized IoT. An optimized IoT will be a sturdy foundation on which security can ride.

3 SYSTEM DESCRIPTION AND MODEL

This section gives a description of the proposed system, highlighting the framework. The intended system is made up of four major entities, where each entity is inter-connected with easy access for all processes. The first entity is the client – comprising of the edge node and the Cluster Heads (CH): the edge nodes are IoT devices that are assigned services, they are made up of varying storage, processing, and computational capacity. The CHs are responsible for sending and receiving tasks/requests, they can communicate with all nodes and local servers; the second entity comprises the service, these are software, that perform required tasks (threads, processes, data flows, etc.), responding to hardware events and/or listens for data requests from other software; the third is the server: they are programs or hardware that provide services and processes that are specified response to requests made by the clients. The CHs and servers form the Dew layer; finally, the cloud makes the fourth and is to some extent intertwined with the third entity as they are remote servers responsible for producing services.

The algorithm regulating the functionality of the proposed system is a hybrid algorithm tagged (RR-RB): unifying a static and dynamic algorithm to allow the best of both worlds, the two unified algorithms are the:

- Static algorithm: Round Robin (RR),
- Dynamic algorithm: Resource based (RB).

The proposed topology is a three-tiered model which minimises the number of connections made by individual nodes. The major layers in a bottom-up fashion are:

- Edge node (IoT devices) layer – first layer: Made up of IoT devices, also referred to as edge nodes. At this layer, user interaction occurs.
- Dew layer – second layer: Made up of designated cluster heads, these cluster heads communicate with each other, edge nodes and servers/resources.
- Cloud layer – third layer: Constituting cloud servers, they communicate with each other and cluster heads as peers.

Figure 2 gives a pictorial representation of the proposed system. Due to the dynamic nature of nodes within an IoT system, the CHs have been carved out to tackle the issue of latency and redundancy by creating and storing paths of frequency and recency, this singular step will abate the unnecessary time required in producing a pathway each time a service is prompted, or a request is made. These cluster heads will also be saddled with the task of adding and removing paths from memory. Following is a breakdown of all the properties and plausible interactions that each entity is charged with. This breakdown has been categorised into three main phases: the classification, where all IoT devices are classified as either edge nodes or cluster heads and activities are classified as either tasks or resources; the resource allocation and scheduling, governed by the hybrid load balancing algorithm; and the communication between all nodes involved.

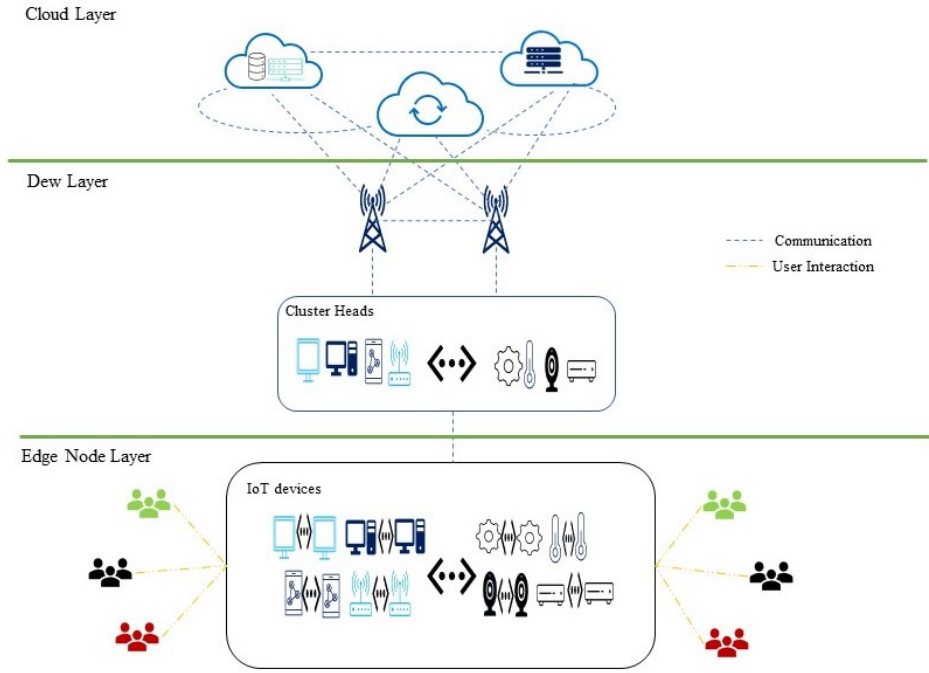

Figure 2. Model architecture

3.1 Classification

Node Classification. At the Edge Node layer, the classification of all nodes simply divides the nodes into two categories: the edge nodes or cluster heads. Assume there are N nodes, let $N = \{n_1, n_2, \ldots, n_{num}\}$ denote the set of all edge nodes in the system where num is the total number of edge nodes and $CH_n \subseteq N$ is the set of cluster heads. At the Dew and Cloud layers: let $S = \{s_1, s_2, \ldots, s_n\}$ denote all servers and $ES_n \subseteq S$ be the set of edge servers. Each node n_i, s_i, es_i has storage capacity St_i and computing power C_i (i.e., the number of hash operations per unit time), where n_i, ch_i, s_i, es_i, st_i, c_i respectively represent the i^{th} node, cluster head, server, edge server, storage capacity and computational power; $i = 1, 2, \ldots, n$.

For a node to be considered the CH, it must, at the time of designation, be free and unsaddled with workload or responsibilities and have high storage and computational capacity in comparison to all other nodes. The CH may or may not be the requester of a given service at a given time. Given the circumstance, the CH is then encumbered with the responsibility of providing the edge node with the necessary information from its archive required for task completion for the node of contention. The CH and the edge servers create the Dew layer, at this layer, the nodes and servers communicate as peers. At the initialization stage, the nodes at

Algorithm 1
1: Initialize: $N_{EdgeNode} = N_{ClusterHead} = \emptyset$,
minimum space required for storing blockchain: st_{min},
time constraint for mining: t_M,
minimum computation required for mining: c_{Mmin};
2: for $n_i \in N$ do
3: if $St_i \geq st_{min}$ and $t_M \cdot C_i \geq c_{Mmin}$ then
4: $n_i \rightarrow N_{ClusterHead}$;
5: else
6: $n_i \rightarrow N_{EdgeNode}$;
7: end if
8: end for
9: Output $N_{EdgeNode}$, $N_{ClusterHead}$

Table 1. Node classification

the Edge Node (first) level can communicate with each other, however, only the CH can interact with the servers, this is to ensure that the traffic is controlled, lighter, and bottleneck is minimal. The CH will be able to key into predetermined paths, centred on the frequency of requests and consequently rate of recurrence of traversing a given path.

Task or Resource Classification. An effective resource allocation strategy needs to comprehensively consider several conditions: the load, network conditions of the server at time t, and QoS parameters in order to effectively realise task classification [12]. In the proposed model, each node n has T processing tasks, defined as $T = \{t_1, t_2, \ldots t_n\}$, assigned to R resources also defined as $R = \{r_1, r_2, \ldots, r_m\}$, where t_i represents the i^{th} task; $i = 1, 2, \ldots, n$ and r_j represents the j^{th} resource; $j = 1, 2, \ldots, m$.

The edge nodes may at a given time make some demands that do not require the facilities of a resource, owing to the fact that such demands at the time of prompting can be handled at the first/Edge Node layer. Therefore, for a demand from an edge node to be considered a 'task', the demand must require the facilities of a resource. On the other hand, for a server to be considered a resource, it must at a given time have all necessary functionalities to meet the demands made by the task.

3.2 Resource Allocation and Scheduling

Load balancing is key for efficiency of the proposed model. "The main reason of introducing any load balancing algorithm is achieving scalability. Basically, it means ability to increase system's performance by extending the cluster with new nodes" [18]. The concept of load balancing views resources as processors, it is not bounded by medium or locality, and services can be accessed globally. Load balancing has two facets: allocation and scheduling. Allocation is usually the focus

of most load balancing algorithms as it is viewed the most important. Resource allocation allows for the addition and removal of virtual servers pending on the resources needed, thereby eliminating problems associated with downtime creation. Scheduling builds on that and assigns resources to tasks as it tackles the problem of which resources needed to be allocated to the received task [19]. Load balancing algorithms can be classified based on four major features: physical location of cluster nodes; visibility in IP network; OSI model layer on which they operate; and static or dynamic character [18].

In the proposed system, the balancing of resources/tasks is structured by the hybrid algorithm, merging the static round-robin (RR) and dynamic resource-based algorithm. By creating this hybrid, not only will there be an active use of feedback from servers to make request distribution decisions, but also an expected low overhead cost. Other advantages of using this hybrid include enhanced performance, resilience, security, and scalability.

Efficient load balancing algorithms take into consideration the total execution time and available resources, ensuring that there is no disruption of service and that there is fast recovery, and extra allowance for fault tolerance. The RR algorithm, when working with resources of equal value, depends on a rotation system to sort traffic, by transferring tasks to the first available server, which is then placed at the bottom of the queue. The resource-based algorithm, on the other hand, harnesses a report of the current task and resource availability to make informative decisions, directing the task to the best suited resource at a given time. The RR-RB therefore aids in obtaining resource optimization.

The total task size TTS generated by N for execution has a transfer rate TR. Q represents the quantum time, that is the time the scheduler allows for a task to run. Assuming that the quantum time of the i^{th} task in the j^{th} resource is $Q(i,j)$, the task size is $\text{TS}(i,j)$, and transfer rate is $\text{TR}(i,j)$, then the turnaround time (TAT) can be expressed as:

$$\text{TAT} = \frac{\text{TS}(i,j)}{\text{TR}(i,j)} \times Q(i,j) \tag{1}$$

where $Q(i,j)$ is defined as:

$$Q = \frac{\text{TTS}(i,j)}{\text{TR}(i,j)}. \tag{2}$$

And TTS is the Total Task Size, that is the summation of all tasks' (ΣT) awaiting resources. During resource allocation, at each node n_i, ch_i, s_i, and es_i, the data size generated by task is prone to data redundancy T_d. The measure of the data redundancy is used

$$T_d = \begin{cases} 1, & \text{if } T_{dlower} < \text{Data}_{node}, \\ 0, & \text{else}, \end{cases} \tag{3}$$

where T_d is the device data redundancy. The upper limit, T_{dupper}, and the lower limit, T_{dlower}, can be obtained from the underlying edge node.

3.3 Communication

At the layers, it is assumed that all nodes (n_i, s_i, and es_i) must communicate with CHs. Therefore, for each case of the load balancing when allocating and scheduling resources, the total cost of communications (T_c) must be calculated.

$$T_c = \frac{\sum_{j=1}^{|V_g|} \left(d_j^r X d^g\right)}{P} \tag{4}$$

where $|V_g|$ is the total number of cluster heads, d_j^r is the total cost of transferring data between i^{th} cluster head and all resources connected to it, d^g is the total cost of communication between cluster heads, and P is the total value of penalty considered for balancing. Equations (5), (6), (7) and (8) further elaborate each sub-component found in Equation (4):

$$d_j^r = \sum_{k=1}^{\left|V_g^j\right|} \varepsilon_{jk} \tag{5}$$

where ε_{jk} is the cost of communication between j^{th} cluster head and all resources connected to it. $\left|V_g^j\right|$ is the number of connected resources to cluster head j.

$$d^g = \sum_{i=1}^{|V_g|} \sum_{\substack{j=1 \\ j \neq 1}}^{|V_g|} l_{ij} \tag{6}$$

where l_{ij} is the cost of communication between servers i and j.

$$P = 1 + \sum_{i=1}^{|V_g|} P_i \tag{7}$$

where P_i is the penalty for the i^{th} cluster head. The value of P_i is calculated as follows:

$$P_i = \begin{cases} 1, & \text{if } g_i^t \leq \epsilon \frac{V_r|}{|V_g|}, \\ 0, & \text{if } g_i^t > \epsilon \frac{|V_r|}{|V_g|}. \end{cases} \tag{8}$$

In which, g_i^t is the number of resources assigned to cluster head i, $|V_r|$ is the number of resources and ϵ is a constant number.

4 ALGORITHM AND IMPLEMENTATION

The following section looks intently at the algorithm and implementation of the proposed model.

4.1 Algorithm of Proposed System

Load balancing algorithms migrate Virtual Machines (VM) between hosts in order to balance host loads. Migrated VMs experience performance degradation which results in lower quality of service (QoS) and can possibly result in Service Level Agreement Violations (SLAV). Hence, an optimal load balancing method should reduce the number of over- and under-utilized hosts with a minimal number of VM migrations [20]. In the proposed model, the host nodes requesting for resources are allocated VMs by a time-shared policy. The use of the RR-RB hybrid algorithm works diligently to achieve optimization by taking into consideration the total execution time and available resources, ensuring that there is no interruption of service and that there is quick recovery, and surplus margin for fault tolerance. The RR algorithm, considered the most optimal algorithm amongst several algorithms, is used for scheduling in the CPU and can be implemented to schedule the processes in real time. RR's attribute of scheduling resources in real time and its ability to allocate requests to a list of resources via the DNS were harnessed in the proposed algorithm, thereby ensuring that all task scheduling occurs in real time and tasks are sent directly to allocated resources. RB, a dynamic load balancing algorithm, on the other hand, allocates load based on what resources each server has available at the given time. Specialized software (called an "agent") running on each server measures that server's available CPU and memory, and the load balancer queries the agent before distributing traffic to that server. RB's ability to allocate resources to tasks dynamically based on available resources at any given time was used and incorporated into the proposed model.

The intended model's ability to apply aspects of both algorithms allows for the allocation of resources to tasks based on resource/server rates and request/task size as well as scheduling of resources in real time, thus eliminating the tendency to under or over utilize resources. Given that, the essence of this system is to tackle scalability, and latency issues that give rise to computational overheads, the structure of the proposed model enables this to be met, whilst taking into consideration QoE and QoS. The RR-RB algorithm significantly reduces, almost eliminates, the vulnerability of resource competition by controlling the number of connections as well as over-utilization of hosts. This step ensures performance interference and subsequently degradation leading to bandwidth congestion, increase in network latency and the likes are tackled. The proposed RR-RB algorithm places equal importance on resource allocation and scheduling; allocation of the tasks to processors, and scheduling of the tasks allocated to a processor are conducted in each case.

Algorithm 2 depicted in Table 2 shows how the proposed system functions, nodes requesting services/resources are allocated resources that have a larger or equal size and capacity for optimal processing. For a resource to be allocated, its size and capacity must surpass or equal the request/task size.

Algorithm 2
1: Initialize: $N = T = R = \emptyset$ Task size: TS, resource size: RS;
2: for $n_i \in N$ do
3: if $r_i \geq t_j$ and $rs_i \geq ts_j$ then
4: $n_i \rightarrow R$;
5: else
6: $n_i \rightarrow i{+}{+}$;
7: end if
8: end for
9: Output: Optimized resource allocation.

Table 2. Proposed system resource allocation algorithm

4.2 Implementation

Recently, multiple work on resource allocation have been put forward and there appears to be a plethora of algorithms to choose from. In a bid to make the simulation as close to reality as possible, the dataset is a collection of random tasks with varying sizes – ranging from 1 to 140, with the number of resources set at 5. This emulates how resource allocation can be manipulated with limited resources. The major parameters that will be used to generate results are time, data transmission rate and size. Table 3 shows a breakdown of the previously mentioned parameters.

Parameter	Metric
Time	Quantum time (ms) Execution time (ms) Turnaround time (ms)
Data transmission rate	Sever rate (kb at every ms) Transfer rate (kbs^{-1})
Size	Task size (kb) Total task size (kb) Resource capacity

Table 3. Parameter-metric

The proposed system was simulated in the MATLAB environment, and it is assumed that the following are known: the number of servers/resources; the processing power of each server; and the request size. By developing this model, the unrealistic expectations, especially for large scale IoT, are resolvable.

Two objects were created: the nodes and servers. The nodes have the attributes: size of requests and turnaround time – time it takes for the request to be managed. The server has the transfer rate, measured in milliseconds (ms) – and the number of requests it can manage at any given time. Each servers' performance will be judged based on how they carried out assigned requests, however, the system's performance will be the turnaround time for all requests processed. Each item in the edge node will be added to a server with iteration set at 20. Each CH interacting with a server

pending on initiated requests of any kind, computes the time it takes for each request to be processed, that is from request initiation to completion of task. The output is collated to checkmate server's performance. This crucial step allows for appropriate allocation of resources, thus an efficient output, making QoS and QoE palpable. The handling of resources by the server via the edge nodes follows a one-to-one mapping and it is expected that any additional technique added to load balancing should reveal a level of improvement.

At the Edge Node (first) layer, the RR-RB algorithm is used to publish how many available resources can be spared at a given time as well as the resource computing power, these information are provided to the CHs by edge nodes after completion of a request. The CH stores and utilizes this information as well as the request/task size from incoming edge nodes for appropriate resource allocation. By declaring the performance level of each server of interest, for instance, how many requests a server can address at any given time and showing the performance of the server in relation to the load handled at a given time as oppose just focusing on the overall performance, the RR-RB algorithm ensures a reduction of overhead costs. If a node is being served, it will have a record of both the best and worst case, this information stored in the CH will be provided.

The three major instances for consideration after implementations are:

- Load balancing based on number of tasks (requests), as depicted in Figure 3: where nodes are assigned servers randomly whilst considering number of tasks awaiting resources;
- Applying load balancing but focusing on the task's size: here, allocation is dependent on the size of the request made by the node, this is portrayed in Figure 4;
- Comparing the results of resources allocated based on number of tasks (Figure 3) and resource allocated based on task size (Figure 4), as shown in Figures 5 and 6.

A fourth instance where results of the proposed model will be compared with an existing algorithm, as depicted in Figures 7 and 8, will be presented.

These simulated instances thereby enable the analysis of data to produce an output of several graphs and charts that show consistent improvement of resource optimization as a result of the simulated load balancing technique as well as the superiority of the proposed model over an existing model.

5 EXPERIMENTS AND ANALYSIS

This section elaborates the results obtained, giving evaluations on the performance of the proposed algorithms under various conditions and setups, as well as comparing with existing works.

5.1 Analysis of Proposed System

The experiment outputs three main test scenarios/selection criterion as well as comparison of existing approaches. All simulated scenarios maintain the same dataset and have a bottom-up approach. After analysis and comparisons are made, suggestions for future work will be proffered.

The output of Figure 3 expresses the simulation imitating the existing system (random allocation), one where any request/task can be assigned to any resource, without any form of optimization technique. Here, each task/request is randomly assigned to a resource/server regardless of the task size and resource capacity. The optimized allocation however, considers the server/resource capacity as well as the number of tasks requiring resources, each node that sends a request will be assigned to an available and appropriate resource. The outcome of Figure 3 shows the immense improvement when aspects of resource allocation are applied. This implies that resources allocation can positively influence a more efficient system.

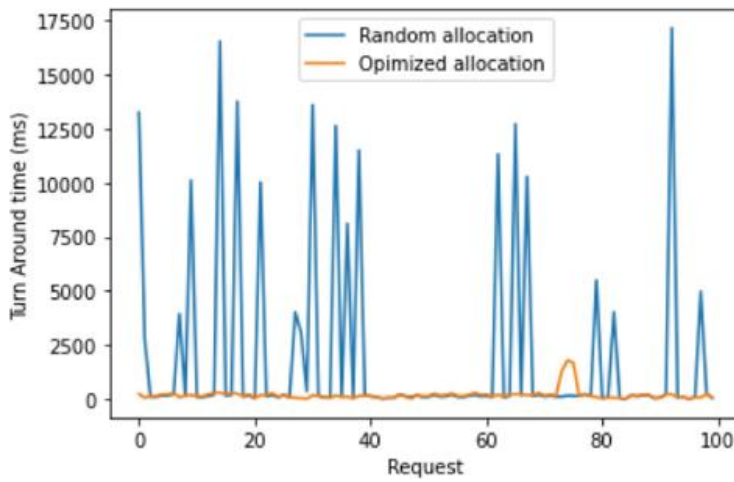

Figure 3. Load balancing based on number of tasks (requests)

Taking a further step to make the optimization techniques even more efficient, the task/request size is taken into consideration in the comparisons made in Figure 4, as oppose Figure 3's approach where the task/request size is not considered. The task/request size measured in bytes is the dimension of the task requiring a resource at any given time. This comparison of random resource allocation and allocation based on request size shows that once the size of the task is known in conjunction with the rate at which a server processes, tasks can be assigned and allocated to an appropriate resource. This is made possible by accessing records stored in the CH of previous turnaround time, paths of frequency and recency, by nodes answerable to the CH, thereby ensuring that resource allocation is balanced and not random. By

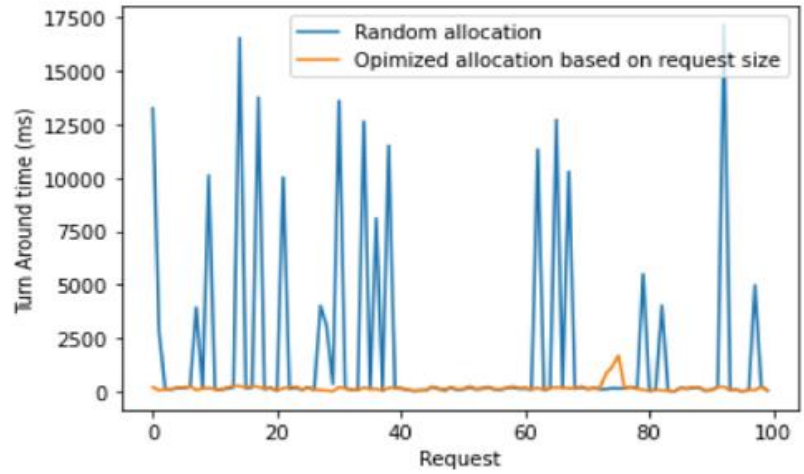

Figure 4. Load balancing based on task (request) size

allocating resources based on task's size, the problem of straining a resource that has higher capacity when there are equally capable resources with lesser capacity is evaded.

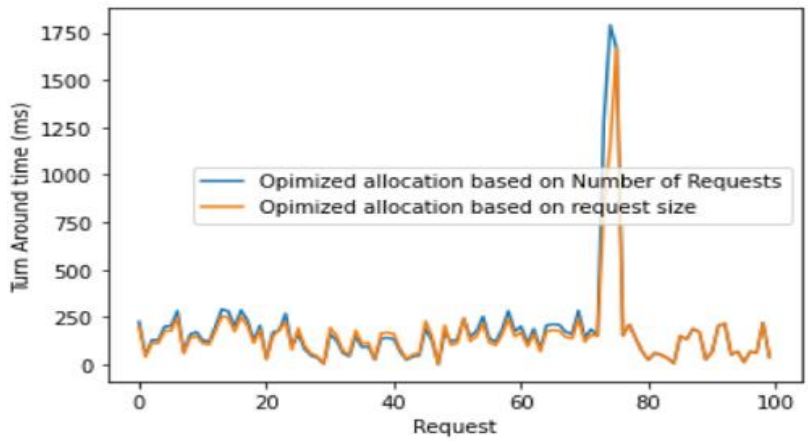

Figure 5. Comparison of load balancing based on number of tasks (requests) and task (request) size

The evident improvement, as depicted in Figures 3 and 4, makes the optimized allocations and the optimized allocation based on request size appear flat planed with TATs closer to zero as oppose the large TAT (17 000 ms) of random allocation.

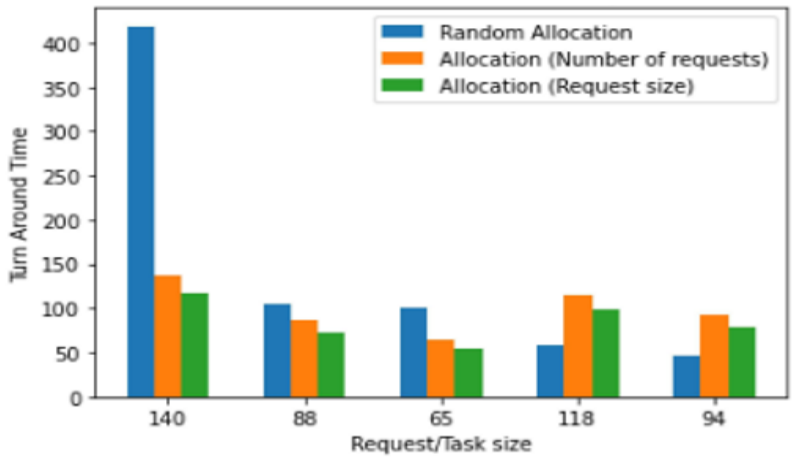

Figure 6. Comparison of load balancing based on random allocation, number of tasks (requests) and task (request) size

The huge gap in TAT of the compared algorithms in Figures 3 and 4 do not give room for a detailed representation of optimized allocations' outputs. However, when compared with more optimized algorithms, as represented in Figures 5, 6, 7 and 8 the evident fluctuations of the optimized approaches are seen, this is due to the TATs of the compared algorithms being significantly closer.

The pictorial representation in Figures 5 and 6 highlights the need to consider the task size over the number of requests when allocating tasks to resources, as it shows how optimization based on request size is relatively more efficient. Consider a scenario where server A has transmission capacity of 200 Gbps and server B has transmission capacity of 100 Gbps, given the capacity of these resources, when optimizing based on number of tasks/requests, there is a tendency to saddle server A with more tasks than server B. This leaves server A strained and prone to down time. However, when optimizing based on task size, the task is assigned to the sever with sufficient capacity, thus resulting in efficient processing and optimized TAT. Although, at certain points, as seen in Figure 6, random allocation produces a better TAT as oppose the optimized algorithms, it suffices to say that the consistency of that occurring, especially for larger task sizes is minimal.

5.2 Comparisons of Proposed System and Existing Systems

Multiple approaches have been put forward as solutions for tackling resource allocation problems in IoT: from genetic algorithm approach to the newer metaheuristic approach. Ant colony optimization, a probabilistic technique for solving nondeter-

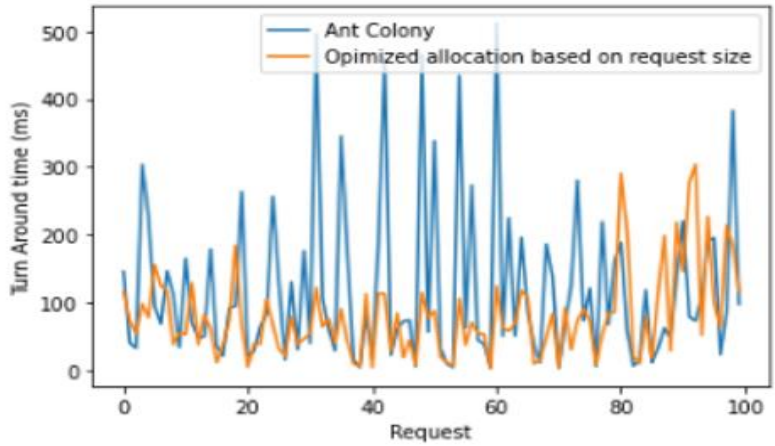

Figure 7. Comparison turnaround time by different methods

ministic problems is a population-based meta-heuristic optimization method that finds shortest paths. The apparent similarities of ant colony to the proposed model, as oppose particle swarm optimization and genetic algorithms, given that the proposed system also tries to find the best/shortest path after harnessing information about paths of frequency and recency stored in the CH led to comparison of the

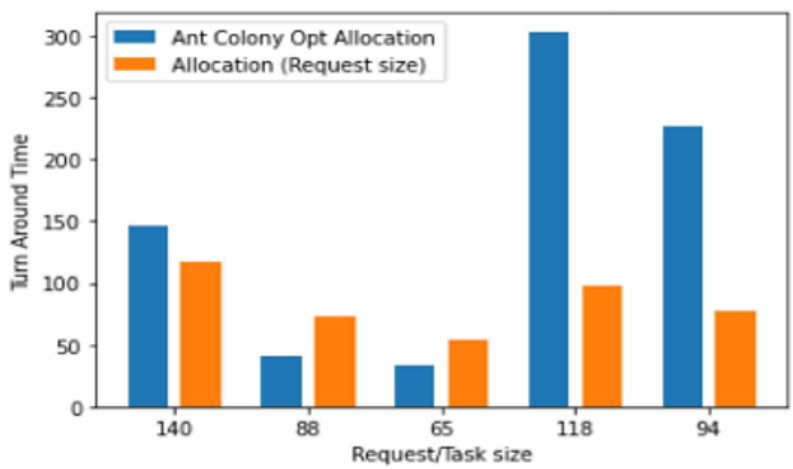

Figure 8. Comparison turnaround time by different methods (bar chart)

proposed system with parameters found in [12]. Employing the parameters of the proposed model and comparing with existing approach in [12] will provide clarification as to the effectiveness of the proposed system.

The pictorial results as indicated in Figures 7 and 8 concomitantly show comparisons of an existing model with the proposed model. From the results, it is evident that after applying the ant colony algorithm on the same dataset used for the proposed system, the proposed system's algorithm largely performs better, especially for larger requests/task sizes. It suffices to say that although certain parameters are controlled, the random nature of task size cannot be controlled, hence, although there is a slight dip in the pattern of TAT at task sizes between 65 and 93, as depicted in Figures 7 and 8, the results reflect a real-life situation. The focus here is on how efficiently these resources are used and how situations such as bottleneck can be avoided. The model so far indicates that resource optimization based on the proposed system is plausible and generally more effective than the ant colony approach. The results further show that having prior knowledge of resource availability and task size have better outcome than allocating resources based on the rate of task completion. In Figure 7, the TAT of both the ant colony and optimized allocation based on request size are closer to zero as oppose that of random allocation, as depicted in Figures 3 and 4, thus the TAT representation is more focused, therefore the apparent fluctuations are noticed.

6 CONCLUSION

Related works indicate that using genetic algorithms produce better solutions, however, this approach does not scale well with complexities. Given that IoT has a tendency to grow in complexity with an increase in size, resultant of its centralised topology, the designed and developed model is proffered to tackle the aforementioned problems by decentralising the IoT topology, thereby drastically reducing complexities. A hybrid approach combining verified server resources and dynamic allocation produces more optimal outputs than the existing ant colony methodology, once the right task is assigned to the right resource, an optimized turnaround time (TAT) is inevitable. An efficient execution time implies an excellent TAT, and an improved TAT would have a waterfall effect on the QoE and QoS. The decentralising of IoT's topology, however, has its own challenges, as the implementation procedure is not straightforward and comes with concerns over processing power and storage capabilities, moreso, security and privacy risks remain a concern, as the network is based on sharing data and tasks between nodes. Given the results presented after implementation of the proposed model, future research can build on the developed model by:

- Utilizing clustering algorithms at the Edge Node layer to further improve the TAT;

- Introducing more optimized concepts that will enhance IoT security; and
- Applying the proposed model to an actual environment for testing.

REFERENCES

[1] Deng, S.—Xiang, Z.—Zhao, P.—Taheri, J.—Gao, H.—Yin, J.—Zomaya, A. Y.: Dynamical Resource Allocation in Edge for Trustable Internet-of-Things Systems: A Reinforcement Learning Method. IEEE Transactions on Industrial Informatics, Vol. 16, 2020, No. 9, pp. 6103–6113, doi: 10.1109/TII.2020.2974875.

[2] Li, X.—Xu, L. D.: A Review of Internet of Things – Resource Allocation. IEEE Internet of Things Journal, Vol. 8, 2021, No. 11, pp. 8657–8666, doi: 10.1109/JIOT.2020.3035542.

[3] Hu, C.—Yang, C.—Fang, X.—Zhi, H.—Hang, L.: A D2D Resource Allocation Method for IoT Network. 2021 IEEE International Conference on Electronic Technology, Communication and Information (ICETCI), 2021, pp. 58–62, doi: 10.1109/ICETCI53161.2021.9563426.

[4] Lieira, D. D.—Quessada, M. S.—Cristiani, A. L.—Immich, R.—Meneguette, R. I.: TRIAD: Whale Optimization Algorithm for 5G-IoT Resource Allocation Decision in Edge Computing. 2021 16th Iberian Conference on Information Systems and Technologies (CISTI), IEEE, 2021, pp. 1–6, doi: 10.23919/CISTI52073.2021.9476599.

[5] Alqerm, I.—Pan, J.: DeepEdge: A New QoE-Based Resource Allocation Framework Using Deep Reinforcement Learning for Future Heterogeneous Edge-IoT Applications. IEEE Transactions on Network and Service Management, Vol. 18, 2021, No. 4, pp. 3942–3954, doi: 10.1109/TNSM.2021.3123959.

[6] Hussain, F.—Hassan, S. A.—Hussain, R.—Hossain, E.: Machine Learning for Resource Management in Cellular and IoT Networks: Potentials, Current Solutions, and Open Challenges. 2019, doi: 10.48550/ARXIV.1907.08965.

[7] Khan, L. U.—Yaqoob, I.—Tran, N. H.—Kazmi, S. M. A.—Dang, T. N.—Hong, C. S.: Edge-Computing-Enabled Smart Cities: A Comprehensive Survey. IEEE Internet of Things Journal, Vol. 7, 2020, No. 10, pp. 10200–10232, doi: 10.1109/JIOT.2020.2987070.

[8] Xu, L. D.—Lu, Y.—Li, L.: Embedding Blockchain Technology into IoT for Security: A Survey. IEEE Internet of Things Journal, Vol. 8, 2021, No. 13, pp. 10452–10473, doi: 10.1109/JIOT.2021.3060508.

[9] Ali, M. S.—Vecchio, M.—Pincheira, M.—Dolui, K.—Antonelli, F.—Rehmani, M. H.: Applications of Blockchains in the Internet of Things: A Comprehensive Survey. IEEE Communications Surveys and Tutorials, Vol. 21, 2019, No. 2, pp. 1676–1717, doi: 10.1109/COMST.2018.2886932.

[10] Zhao, H.—Deng, S.—Zhang, C.—Du, W.—He, Q.—Yin, J.: A Mobility-Aware Cross-Edge Computation Offloading Framework for Partitionable Applications. 2019 IEEE International Conference on Web Services (ICWS), 2019, pp. 193–200, doi: 10.1109/ICWS.2019.00041.

[11] Zhang, Z.—Zhang, W.—Tseng, F. H.: Satellite Mobile Edge Computing: Improving QoS of High-Speed Satellite-Terrestrial Networks Using Edge Computing Techniques. IEEE Network, Vol. 33, 2019, No. 1, pp. 70–76, doi: 10.1109/MNET.2018.1800172.

[12] Qiao, Z.: Research on Optimization Algorithm of Cloud Computing Resource Allocation for Internet of Things Engineering Based on Improved Ant Colony Algorithm. Mathematical Problems in Engineering, Vol. 2022, 2022, Art. No. 5632117, doi: 10.1155/2022/5632117.

[13] Minovski, D.—Åhlund, C.—Mitra, K.: Modeling Quality of IoT Experience in Autonomous Vehicles. IEEE Internet of Things Journal, Vol. 7, 2020, No. 5, pp. 3833–3849, doi: 10.1109/JIOT.2020.2975418.

[14] Suryanegara, M.—Prasetyo, D. A.—Andriyanto, F.—Hayati, N.: A 5-Step Framework for Measuring the Quality of Experience (QoE) of Internet of Things (IoT) Services. IEEE Access, Vol. 7, 2019, pp. 175779–175792, doi: 10.1109/ACCESS.2019.2957341.

[15] Dong, L.: A New Dynamic Load Balancing Algorithm for Multi-ROIA. Computing and Informatics, Vol. 38, 2019, No. 1, pp. 1–18, doi: 10.31577/cai_2019_1_1.

[16] Mishra, N.—Pandya, S.: Internet of Things Applications, Security Challenges, Attacks, Intrusion Detection, and Future Visions: A Systematic Review. IEEE Access, Vol. 9, 2021, pp. 59353–59377, doi: 10.1109/ACCESS.2021.3073408.

[17] Lepekhin, A.—Borremans, A.—Ilin, I.—Jantunen, S.: A Systematic Mapping Study on Internet of Things Challenges. 2019 IEEE/ACM 1st International Workshop on Software Engineering Research and Practices for the Internet of Things (SERP4IoT), 2019, pp. 9–16, doi: 10.1109/SERP4IoT.2019.00009.

[18] Opioła, Ł.—Dutka, Ł.—Wrzeszcz, M.—Słota, R.—Kitowski, J.: Two-Layer Load Balancing for Onedata System. Computing and Informatics, Vol. 37, 2018, No. 1, pp. 1–22, doi: 10.4149/cai_2018_1_1.

[19] Madni, S. H. H.—Latiff, M. S. A.—Coulibaly, Y.—Abdulhamid, S. M.: Resource Scheduling for Infrastructure as a Service (IaaS) in Cloud Computing: Challenges and Opportunities. Journal of Network and Computer Applications, Vol. 68, 2016, pp. 173–200, doi: 10.1016/j.jnca.2016.04.016.

[20] Moghaddam, S. M.—O'Sullivan, M.—Unsworth, C. P.—Piraghaj, S. F.—Walker, C.: Metrics for Improving the Management of Cloud Environments – Load Balancing Using Measures of Quality of Service, Service Level Agreement Violations and Energy Consumption. Future Generation Computer Systems, Vol. 123, 2021, pp. 142–155, doi: 10.1016/j.future.2021.04.010.

Dorcas Dachollom Datiri is Doctoral Researcher at the Department of Electronic and Electrical Engineering, Brunel University, London, UK. She is also Graduate Teaching Assistant at the same university. She received her Bachelor's degree in computer science from the Bingham University, Nigeria in 2010 and her Master of Science degree in advanced computer science from the University of Leicester, UK in 2015. She has been Lecturer at the University of Jos, Nigeria (2015–2019). Her research interests are data science, data analytics, optimization techniques, edge and cloud computing, blockchain technologies, Internet of Things and Industry 4.0. She is an Associate Fellow in Higher Education, UK, and member of the ISOC and NACOSS professional bodies.

Maozhen Li is Professor in the Department of Electronic and Electrical Engineering, Brunel University, London, UK. He received his Ph.D. from the Institute of Software, Chinese Academy of Sciences in 1997. He was Post-Doctoral Research Associate in the School of Computer Science and Informatics at Cardiff University, UK in 1999–2002. His main research interests include high performance computing, big data analytics and intelligent systems with applications to smart grid, smart manufacturing and smart cities. He has over 200 research publications in these areas including 4 books. He has served over 30 IEEE conferences and is on the editorial board of a number of journals. He is Fellow of the British Computer Society (BCS) and the Institute of Engineering and Technology (IET).