Computer Systems
Science and Engineering

Tech Science Press

DOI: 10.32604/csse.2023.037748
Article

Edge of Things Inspired Robust Intrusion Detection Framework for Scalable and Decentralized Applications

Abdulaziz Aldribi[1,2,*], Aman Singh[2,3] and Jose Breñosa[3,4]

[1]Department of Computer Science, College of Computer, Qassim University, Buraydah, P.O. Box 52571, Saudi Arabia
[2]Prince Faisal binMishaal Artificial Intelligence Chair, Qassim University, Buraydah, P.O. Box 52571, Saudi Arabia
[3]Higher Polytechnic School, Universidad Europea del Atlántico, C/Isabel Torres 21, Santander, 39011, Spain
[4]Department of Project Management, Universidad Internacional Iberoamericana, Campeche, C.P. 24560, Mexico
*Corresponding Author: Abdulaziz Aldribi. Email: aaldribi@qu.edu.sa
Received: 15 November 2022; Accepted: 02 February 2023

Abstract: Ubiquitous data monitoring and processing with minimal latency is one of the crucial challenges in real-time and scalable applications. Internet of Things (IoT), fog computing, edge computing, cloud computing, and the edge of things are the spine of all real-time and scalable applications. Conspicuously, this study proposed a novel framework for a real-time and scalable application that changes dynamically with time. In this study, IoT deployment is recommended for data acquisition. The Pre-Processing of data with local edge and fog nodes is implemented in this study. The threshold-oriented data classification method is deployed to improve the intrusion detection mechanism's performance. The employment of machine learning-empowered intelligent algorithms in a distributed manner is implemented to enhance the overall response rate of the layered framework. The placement of respondent nodes near the framework's IoT layer minimizes the network's latency. For economic evaluation of the proposed framework with minimal efforts, EdgeCloudSim and FogNetSim++ simulation environments are deployed in this study. The experimental results confirm the robustness of the proposed system by its improvised threshold-oriented data classification and intrusion detection approach, improved response rate, and prediction mechanism. Moreover, the proposed layered framework provides a robust solution for real-time and scalable applications that changes dynamically with time.

Keywords: Internet of Things (IoT); Edge of Things (EoT); fog computing; cloud computing; scalable; decentralized

1 Introduction

The internet is quickly developing toward the "Internet of Things" (IoT), which has the potential to network billions or possibly trillions of objects. By 2025, more than 50 billion Electronics gadgets and smart devices will be connected to the internet [1]. Most of these devices will be situated near the

internet's edge. They may offer new applications to change various areas of conventional industrial production and daily life. As IoT devices have low computational power and low storage capacity, Cloud services are deployed for complex computation and storage of data. On the other hand, the latency in cloud services is very high, making it less suitable for real-time applications requiring rapid response, like automated automobiles and healthcare sectors [2]. To overcome the consequences of both technologies, Edge Computing (EC) is deployed in various mission-critical applications. EC has created a position in the technical world due to its remarkable performance of providing real-time data processing, cheap operational cost, high scalability, reduced latency, and enhanced quality of service (QoS) [3]. Due to its extraordinary processing power, EC will change several industries, including e-commerce, social networks, agriculture, healthcare, education, and transportation [4]. The centralized approach of processing and storing data is no longer adequate for handling the billions of devices functioning in different geographical locations with dynamic movability.

Real-time and dynamic challenges are best addressed by decentralized, scalable applications currently in demand. EC is a comparably better solution for handling decentralized and scalable applications than the cloud because of its centralized nature [5]. The various challenges IoT and Cloud technology face in addressing issues, including poor network latency, poor performance, and poor stability in the dynamic environment in decentralized and scalable applications, this study proposes an EC-inspired framework that can handle numerous matters comparably better than these technologies separately. The following are the main motivations behind this study:

- Check the various architectures used for implementing edge computing and its integration with other cutting-edge technologies.
- Design a suitable algorithm to decrease the latency present in IoT and cloud computing in decentralized and scalable applications.

The main contributions of this study are centered on the following three points:

- We designed an Edge of Things (EoT)-inspired robust architecture for ubiquitous data monitoring and improvised inter-device communication with minimal response time.
- The work integrates a threshold-oriented robust intrusion detection system in decentralized and scalable applications by incorporating intelligent local nodes at each level of the network hierarchy.
- The study improves the reliability and performance of decentralized and scalable applications by implementing a threshold-oriented, load-balanced, multilayered EoT framework.
- The experimentation also exported various unseen utilization of EoT in decentralized and scalable applications.

The organization of this paper is presented in Fig. 1. Section 2 presents the various architectures used to implement EoT. Further, Section 3 presents the concerns and refinements in scalable and distributed Applications. Subsequently, Section 4 presents the EoT-inspired proposed framework for real-time and scalable applications. Furthermore, Section 5 introduces the material and methods for implementation. Section 6 explores the results obtained and finally, Section 7 concludes the study.

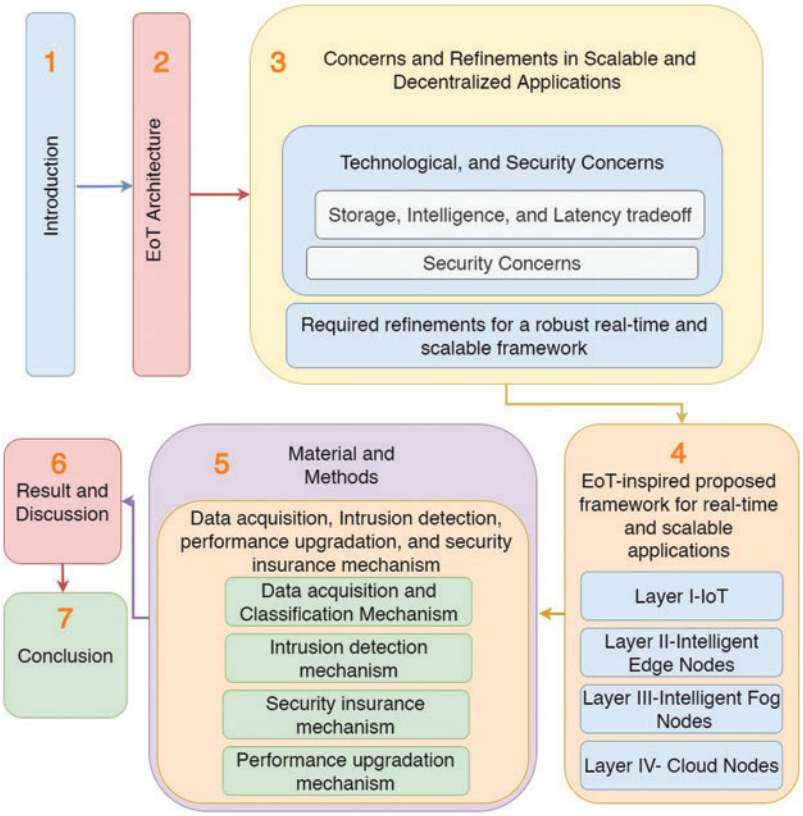

Figure 1: Research progression and milestones

2 EoT Architecture

A taxonomy of EoT Architecture based on the various technological issues raised due to the integration of these two robust technologies covers Data Placement based architectures, Big Data Analysis based architectures, Security based architectures, Machine Learning implementation-basedsed architectures, and Orchestration based architectures. This section illustrates the various EoT Architectures deployed based on the issues mentioned earlier. Table 1: "*EoT (Edge of Things) computing architectures taxonomy*" shows multiple architectural aspects of most of the EoT architectures used in various EoT-based applications. Firstly, in decentralized and scalable applications, the IoT devices rate massive amounts of data in these applications. Placing data on different IoT devices and their respective edge nodes with minimal latency is challenging. To overcome the issues of high latency, IFogStor architecture is deployed. Single integer programs, Divide-and-conquer heuristic approach, Geographical mapping matrices, and the Multi replica Data-Placement are the main strategies used by these architectures respectively to reduce latency in the IoT communication network.

Further, based on Big Data Analysis, Hierarchical Distributed Fog Computing architecture is deployed. Big Data Analysis is required in large-scale applications like smart cities and Industrial IoT. Hierarchical Distributed Fog (HDF) Computing architecture is deployed to manage various Big Data Analysis issues. Further, in this sequence, to handle various security issues in different networks, various Edge computing architectures are deployed. Generally, Privacy Preservation Aggregating (P2A), Lightweight Security Virtualization (LSV), Service Balance Dynamics Based on Cloud (SBDC), and

Edge and Cloud Virtualization (ECV) are deployed to maintain security in network communication. Specifically, in Software Defined Networks (SDN), Software-Defined Fog-Node-Based Distributed Blockchain Cloud Architecture (SDNDB) [6]. Orchestration is a significant characteristic-based taxonomy of edge computing architecture. Most of the architecture uses Software-Defined Networks, whereas the other techniques are suitable for managing various network and resource allocation-related issues. Finally, in this sequence, for Software Defined Network Based Fog Computing Architecture, Vehicular Ad hoc Network (VANET) Architecture, Software Defined Fog-Computing Network Architecture for IoT (SDFN), SDN-Based Cloudlet Architectures, i.e., Dynamic Distribution of IoT Analysis (DDA) are deployed to handle various issues in software-defined networks [7]. Consequently, the following section illustrates the decentralized and scalable approach for a real-time and dynamic environment. Table 1 precisely illustrates the various aspects of various EoT architectures including Blockchain-SDN for Vehicle (BSDNV), SIOTOME, Mobile Crowd Sensing (MCS), Transferring Trained Models (TTM), Hierarchical Fog-Assisted Computing Architecture (HiCH), Vehicles as Fog Computing Infrastructures (VISAGE) [8], Fog Software Defined Network (FSDN), Multi-Agent-Based Flexible Edge Computing Architecture (MAFECA) [9], Mobile Fog Service Allocation (MFSA), Hierarchical Architecture for Mobile (HAM), Scalable IoT Architecture Based on Transparent Computing (SAT), Edge-Based Assisted Living Platform for Home Care (E-ALPHA) [6].

Table 1: EoT (Edge of Things) computing architectures taxonomy

EoT architecture	IFogStor	IFogStorZ	IFogStorG	IFogStorM	HDF	P2A	LSV	SBDC	ECV	SDNDB	BSDNV	SIOTOME	MCS	TTM	HiCH	DDA	VISAGE	FSDN	SDFN	MAFECA	MFSA	HAM	SAT	E-ALPHA
Data placement	√	√	√	√	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Big data analysis	X	X	X	X	√	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Security	X	X	X	X	X	√	√	√	√	√	√	√	√	X	X	X	X	X	X	X	X	X	X	X
SDN security	X	X	X	X	X	X	X	X	X	√	√	√	√	X	X	X	X	X	X	X	X	X	X	X
Machine learning	X	X	X	X	X	X	X	X	X	X	X	X	X	√	X	X	X	X	X	X	X	X	X	X
Orchestration	X	X	X	X	X	X	X	X	X	X	X	X	X	X	√	X	X	X	X	X	X	X	X	X
SDN orchestration	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	√	√	√	√	X	X	X	X	X
Cloudlet SDN	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	√	X	X	X	X	X	X	X	X
Fog SDN	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	√	√	√	X	X	X	X	X
Service and tasks allocation	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	√	√	√	√	√

3 Concerns and Refinements in Scalable and Distributed Applications

In the above section, distributed and scalable applications are described. This section explores various concerns present in real-time scalable and distributed applications, along with the possible refinements by technology integration and architectural improvement.

3.1 Technological and Security Concerns

Technological, security and architectural concerns are the most vibrant concerns observed in real-time applications. This sub-section covers all these concerns sequentially.

3.1.1 Storage, Intelligence, and Latency Tradeoff

In any real-time and scalable applications, billions of internet-connected devices acquire data from the environment. Most of these applications require real-time intelligence decision-making capabilities at the device level or on devices near the user end, like Fog or Edge Computing devices. IoT devices,

including sensors and actuators, have low computational and storage capacity but are good in latency. Cloud devices have high computational and storage capacity but high latency, making them slower for real-time applications. The latency of any network is measured in milliseconds (ms) and is also known as RTT, i.e., Round Trip Time. The RTT of any network mainly depends on Processing Delay (PD), Queuing Delay (QD), Transmission Delay (TD), and Progression Delay (PRD) [10]. Processing Delay in a network depends on various device-level activities, including data acquisition, classification filtering, error detection, next-hop analysis, encryption operations, etc. All processing delays generally depend on user-level devices' storage and computational capacities.

$$TD = \frac{Number\ of\ transmited\ bits{-}NB}{Transmission\ Rate{-}TR} \tag{1}$$

$$PRD = \frac{Distance\ between\ tranmiter\ and\ receiver\ nodes{-}Dis}{Sped\ of\ the\ packet{-}Spd} \tag{2}$$

The overall network latency depends upon all the above mentioned factors formulated under Eq. (3).

$$N_{latency} = PD + QD + TD + PRD \tag{3}$$

$$N_{latency} \propto Dis \tag{4}$$

The discussion, as mentioned earlier, concluded in Eq. (4) that the network latency is directly proportional to the distance. Therefore, we can improve the response time by minimizing the distance between two nodes. So the tradeoff between distance and latency has been observed during this study. Further, the intelligent nodes require storage space because lots of data is necessary to make an intelligent decision. In most of the architectures, all the decision-making capabilities are reserved with cloud nodes as they have huge storage space and can make a decision intelligently. But due to the considerable distance between user respondent nodes and cloud nodes, the response rate became poor in the case of real-time applications. Furthermore, a reasonable response rate is required for handling real-time challenges like intelligent traffic control in smart city networks. Therefore, by reducing the distance between respondent nodes and intelligent decision-making nodes; and making the system decentralized, remarkable improvement in response rate can be possible.

$$ID_{Real-Time} \propto Dis_{R-node\ to\ I-node} \tag{5}$$

$$Where\ ID_{Real-Time} = Real - Time\ Intelligenet\ Decision$$

$$Dis_{R-node\ to\ I-node} = Distance\ between\ Respondent\ Node\ and\ Intelligent\ Node$$

$$ID_{Real-Time} \propto Storage\ and\ computation\ capacity\ of\ Intelligent\ Node \tag{6}$$

3.1.2 Security Concerns

Any real-time and scalable applications require identifying vulnerable areas in a network and real-time intelligent corrective action. Some intelligent mechanisms near the respondent nodes are needed to identify vulnerable areas. In a centralized approach, the respondent nodes, i.e., IoT devices, have less computation and storage capacities, and identifying vulnerable areas is the cloud-level concern. Further, the cloud nodes also decide the required corrective or preventive action. However, this approach is suitable for small and non-scalable, centralized applications. But, in the case of real-time and scalable applications, a robust intrusion detection mechanism is required as multiple security

breaches can be possible over real-time. Therefore they should be safeguarded as part of a system with different levels of IoT safety. Inconvenience is caused by connectivity issues, device issues, or an increase in service assaults. There have been numerous incidents where threats to public safety have resulted in injuries. Another issue is rogue and counterfeit IoT devices. The ability to prevent a system from being accessed from every device is a serious security challenge. Using the IoT in one's house presents a hurdle because it might grow bulky if more home devices are installed. Users frequently install rogue and inferior IoT without authorization on secured networks. These devices merge into the network or replace the existing network units to get private information and data, breaching the network's security perimeter [11]. The IoT device known as a programmable controller was the target of the attackers. All it took for an attacker to compromise the system and expose the internal network was for one worker to insert a micro flash drive into the controller.

3.2 Required Refinements for a Robust Real-Time and Scalable Framework

The nature of real-time and scalable applications is different from static applications. Therefore, the framework for such applications also required refinements for better and more accurate results. This sub-section covers the essential advancements needed for real-time and scalable applications. As in scalable application, the number of sensing devices dynamically vary with time. The respondent nodes must be intelligent enough to make the decisions in a real-time scenario. The decentralized approach is required to overcome various Big-Data and other issues in the centralized system [12]. The intrusion detection mechanism must be employed near the respondent nodes in a decentralized manner. The corrective action for any detected irregularity in a real-time framework must be as quick as possible to minimize the consequences in the framework [13]. Multilayered architecture is required for the refinement of data and improved decision-making powers of the overall framework. The subsequent sections of this study illustrate the proposed framework for real-time and scalable applications incorporating all these refinements.

4 EoT-Inspired Proposed Framework for Real-Time and Scalable Applications

The real-time and scalable applications have some significant characteristics, including better real-time data acquisition mechanisms, refined data classification mechanisms, quick response for every query which means minimal latency, intelligent respondent nodes, intelligent intrusion detection mechanism, and immediate corrective action for minimal damage. The subsequent subsections of this section propose a multilayered framework for real-time and scalable applications, which constitutes of the following layers: Layer I-IoT, Layer II-Intelligent edge nodes, Layer III-Intelligent fog nodes, Layer IV-Cloud nodes. The proposed framework is conceptualized in Fig. 2. It represents two virtually connected geographical areas, that are different geographical areas because of their different geographical presence but the same is virtually connected with each other due to the same network structure and acceptance of dynamically moving objects in both areas.

This study considers two case studies for a better understanding of the real-time, dynamic and scalable scenario. Case Study-I implements the proposed framework in driverless cars, whereas Case Study-II implements this framework in a smart city network. Both of these case studies depict real-time and scalable environments. The mentioned case studies are the most familiar and vibrant applications of real-time and scalable scenarios and understanding the existing technical consequences in these scenarios is easier than in comparison with the other scenarios. The subsequent sub-sections will illustrate all these layers and their respective implementation in these two case studies. Firstly, IoT devices are deployed for data acquisition in real time. Various types of IoT devices, sensors, and

actuators are deployed to sense data. In Case Study-I, the driverless cars deployed multiple sensors, including vision, sound, distance, temperature, humidity, smoke, and more [14]. In Case Study-II, humidity sensor, temperature sensor, light sensor, and many other sensors as per the facilities provided by the smart city network [15]. All these sensors are connected through the internet and can acquire real-time data and communicate with other devices. Moving next in this sequence on layer-II, all the local IoT devices are connected to their respective Edge Nodes. IoT devices have less storage capacity and less computation power. So IoT devices can't make intelligent decisions in real-time. The idea of the placement of Intelligent Edge Nodes near the Local IoT devices reduced the distance of respondent nodes. Most of the data is handled and processed by these edge nodes. In Case Study-I, Image denoising, Image segmentation, Image recognition, Image restoration, Image compression, etc., are various intelligent decisions that the local Edge Nodes can make. Whereas in Case Study-II, Humidity analysis, Electricity consumption, water consumption, etc., are implemented by the Intelligent Edge Nodes.

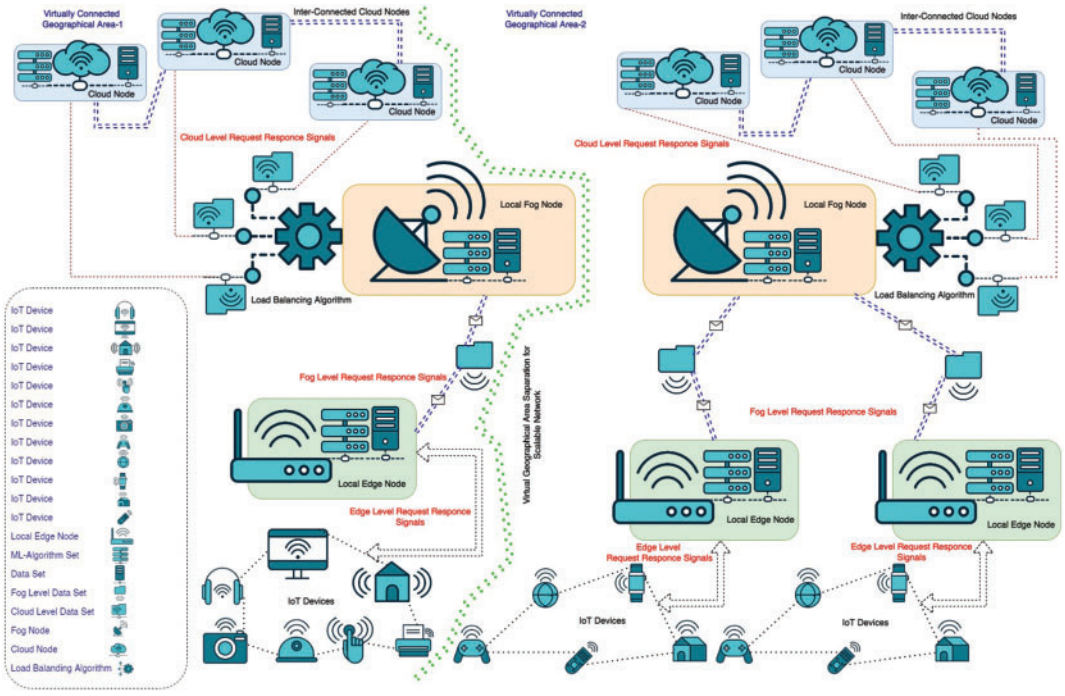

Figure 2: EoT-inspired proposed framework for real-time and scalable application

Subsequently, on layer-III, Numerous Edge Nodes are connected with a single Fog Node. The number of Fog Nodes in a network depends upon the application's complexity and the network's geographical location. Further, the edge devices can dynamically change their respective Fog Node, as per their location and availability of Fog Node. These Fog Nodes are deputed into a network to serve as secondary cloud nodes. Like cloud nodes, these fog nodes have large memory space and computation powers, but with cloud nodes, they have fewer memory and computation powers. In Case Study-I, if the driverless car needs to know the traffic status in a particular place, the request is forwarded to the fog node with the help of connected edge nodes. And the request is served by the respective Fog Node instead of Edge Node. In Case Study-II, the smart homeowner needs to know the electric bill's status and predict the monthly electricity consumption based on previous data stored in fog nodes.

Then these requests are also served by respective Fog Nodes instead of Edge Nodes. Further, in the proposed framework, the fog nodes contain information about all the available cloud nodes and their nature of duty. The fog nodes are also deployed to forward cloud-level requests to the respective cloud node after analyzing the nature of the request. Finally, on layer-IV, each Fog Node is further connected with all the available Cloud Nodes. The number of Cloud nodes in a network depends upon the number of available Fog Nodes and the geographical location of the network. As the proposed system is a distributed system, the functioning of all the cloud nodes is different, depending upon the nature of the application. All the cloud nodes have massive data storage and computation powers. All the cloud nodes are dedicated servers, a single type of request only. In Case Study-I, in the previous example, due to heavy traffic in between, the car needs to know the alternate shortest path to reach its destination on time. Such kind of requests, requires lots of computation power. Therefore, such requests have been handled by cloud nodes. In Case Study-II, if the electricity department needs to calculate the monthly electricity bills of all the users, then such requests need to be served by cloud nodes.

The central idea behind this framework is to segregate all types of requests based on their complexity and allowed response type and handle requests accordingly. All the requests are divided based on two factors i.e., Complexity and Required Response Rate. Various levels of requests are mentioned in Table 2: *"Request Level Table for real-time and scalable applications"*. All Level-I requests are either handled by Edge Node or Fog Nodes but can't be managed by Cloud Node. Further, all Level-II requests can be handled by all three nodes depending upon their complexity and response rate. All Level-III requests are handled by either the Fog node or the cloud node.

Table 2: Request Level Table for real-time and scalable applications. L-Low, M-Medium, H-High, √-can handle by the respective node, X-can't handle by the individual node

Complexity	Required response rate	Level	Edge node	Fog node	Cloud node
L	L	Level-I	√	√	X
L	M	Level-I	√	√	X
L	H	Level-I	√	X	X
M	L	Level-II	X	√	√
M	M	Level-II	X	√	X
M	H	Level-II	√	√	X
H	L	Level-III	X	√	√
H	M	Level-III	X	√	√
H	H	Level-III	X	X	√

5 Material and Methods

This section illustrates various materials and methods used to implement and experiment with the proposed framework. For proper implementation of the suggested framework variety of devices are required. Studying the proposed framework in a real-time scenario is very time-consuming, cumbersome, and expensive. To simplify the implementation and experimentation of the proposed framework, the EdgeCloudSim simulation environment is deployed to make the experimentation part simple and inexpensive. The bandwidth, RAM, CPU, and storage resources in CloudSim are constrained when building a Virtual Machine (VM). There is no restriction on how many tasks can

run on VMs regarding functions. Said, task execution takes longer when there are more tasks to do. We want the edge servers to handle incoming jobs in a short period. Hence this concept of execution time is incompatible with the edge computing methodology. As a result, we built a new CPU utilization model for the VMs that caps the number of concurrent processes that can be run on each VM. Although a single task's CPU usage is static and specified in the configuration file, a dynamic utilization model can also be developed by modifying the relevant CPU utilization class.

Further, for the implementation of fog nodes, FogNetSim++ is deployed. A fog simulator application called FogNetSim++ offers users thorough configuration options to model a sizable fog network. It is built on top of OMNeT++, a discrete event simulation tool that is open source and offers an extensive library for simulating network features. A traffic control system is tested to show the FogNetSim++ simulator's scalability and efficiency in terms of CPU and memory utilization. Network factors such as execution delay, packet error rate, and latency are benchmarked by the authors. FogNetSim++ does not, however, yet facilitate VM movement between fog nodes. Raspberry Pi and OctaCore 64-bit ARM processors (1.4 GHz) were utilized for the simulation environment to deploy the fog computing node. It made use of a 16 GB RAM VideoCore IV GPU. Like this, Windows Server 2016 was installed on an Amazon EC2 machine for a cloud computing platform with 8GB RAM and 4GB SSD. The subsequent subsections of this section illustrate the experimentation environment and various algorithms involved in implementing the proposed framework. The flow of various methods processed during implementations is demonstrated in Fig. 3. In nutshell, in this proposed experimentation scenario, various types of requests are classified based on their required response rate and complexity. Further, the request handling methods are also placed over different levels of the proposed framework. After identifying the request, it is transmitted with its respective request handling level. Such a load-balanced framework helps us to improve the overall response rate of the network. The subsequent sub-sections elaborate on various components and methodologies deployed over the proposed experimental framework.

Data Acquisition, Intrusion Detection, Performance Upgradation, and Security Insurance Mechanism are mainly covered under this section. Including these main components, this section also covers various improvised mechanisms used in the proposed frame for making the proposed system robust for real-time scalable applications. For improvisation, this study presents the following primary mechanism for the overall improvement in the performance of the proposed framework: Data acquisition and classification mechanism, Intrusion detection mechanism, Security insurance mechanism, and Performance upgradation mechanism. The subsequent subsections will explain each of these mechanisms in detail. For better understanding, the previously mentioned case studies, i.e., Case Study-I (Driverless Cars) and Case Study-II (Smart City Network) have been reconsidered.

5.1 Data Acquisition and Classification Mechanism

IoT and Edge devices are deployed for data acquisition in both case studies. IoT devices can efficiently acquire real-time data. The proposed system implemented data classification-based associated methods. All the data obtained by various IoT devices are classified by their respective edge device based on the implementation node of data. It can be possible that the segregated data may be processed or used by multiple nodes and at different levels of processing. The respective edge device is also responsible for checking the functioning of their connected IoT devices. This framework recommends forming a data segregation table before implementing the data classification algorithm. Algorithm 1 illustrates the proposed classification mechanism. Based on the required response rate and complexity of the request all the requests are already segregated in levels and the entries of the

same are already mentioned in the segregation table. Algorithm 1 uses this segregation table for the classification of data.

Algorithm 1: Data Classification Algorithm

```
1   Div_id ← DeviceID, E_id ← EdgeID,
    LF_id ← LocalFogNodeID,
    Div_Data ← DeviceData,
    M_Set ← MethodSet,
    L_Status ← LevelStatus,
    C_id ← CloudID,
    M_id ← MethodID
    Req_id ← RequestID
2   if Req_id = active then
3       Get Div_id
4       Get Div_Data(Div − id)
5       while M_Setdo
6           Check L_Status
7           while L_Status = available do
8               Check Div_id(L_Status)
9               if Div_id(L_Status) = LF_id
10                  Communicate Div_DatatoLF_id
11              else
12                  Check C_id(L_Status)
13                  Communicate Div_DatatoC_id
14              end
15          end
16      end
17  end
18  end
```

5.2 Intrusion Detection Mechanism

Every data item has some threshold values; based on threshold values, vulnerable situations can easily be detected [5,16]. The proposed framework suggested the threshold basis Intrusion Detection System. The respective threshold values have already been saved in their corresponding Edge Device for every chunk of acquired data from a real-time environment. Before implementation of algorithm-1, the respective Edge Node check its threshold limits. If the value is between the threshold limits, algorithm-I is called for necessary action. In Case Study-I, if the Engine Heat Level increases from the specified limit, that indicates a vulnerable situation, and corrective action, i.e., stop the engine, must be executed immediately. The vulnerable situation may occur at each level of the proposed framework. Therefore, Intrusion detection needs to be done at each level. In Case Study-II, if the electricity consumption of an entire month of a house is moving beyond the specified threshold limits, it indicates some issue in one of many appliances in the home. This type of vulnerable situation can be detected at fog level. Algorithm-II illustrates the threshold-based Intrusion Detection Mechanism. The classified data is further analyzed for intrusion detection based on their allowed threshold values already stored in their respective threshold database.

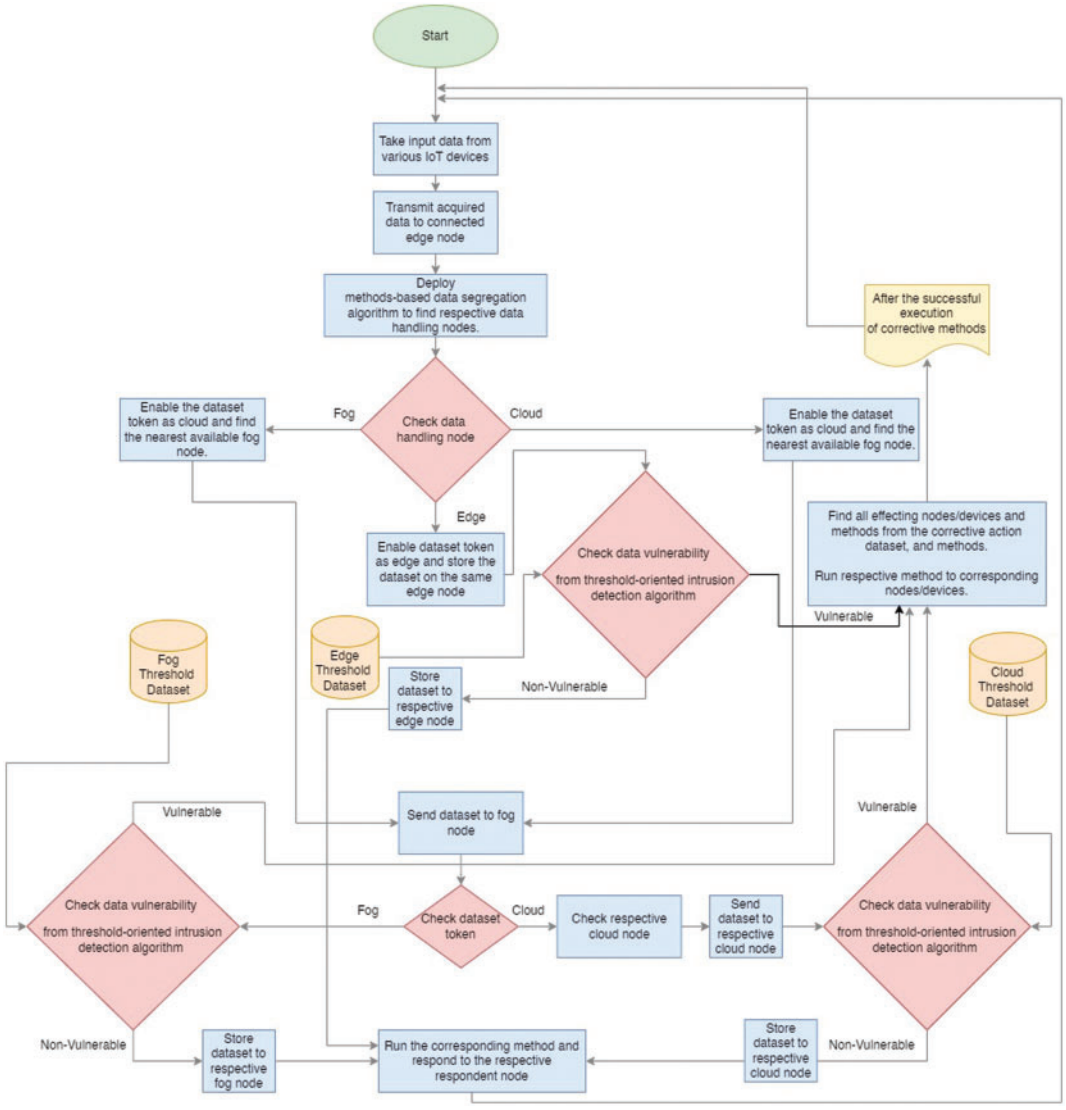

Figure 3: Data flow diagram of the proposed framework

5.3 Security Insurance Mechanism

Security in any framework will only be ensured by predicting vulnerable situations in advance and taking timely corrective action to minimize the damage made by any vulnerable situation. Integrating Edge nodes in between the Fog and IoT devices serves many security issues. For any vulnerable situation, there must be corrective action. All the corrective actions required are also stored along with the threshold limits of data values. As algorithm II is implemented at each level of the proposed framework, the security insurance algorithm will also be implemented similarly. All the issues detected at level-I need to be corrected at a similar level, i.e., level-I. Likewise, all the issues detected by fog and cloud nodes must be addressed only by the respective detected node. Based on the threshold, each vulnerable situation has two corrective actions. The helpless situation may occur either due to exide the value of a specific variable beyond the specified limit or a fall of value below the specified limit. For

example, in Case Study-I, the engine temperature moves down below the specified limit, which may cause trouble in starting the engine, so different corrective action must be taken. On the other hand, the high temperature may damage some other internal parts of the engine, so one must immediately stop the engine. Similarly, two different corrective actions must be performed in all vulnerable cases. Algorithm III illustrates the Security mechanism proposed in this framework. All the corrective actions associated with respective irregularity detected by Algorithm II are stored at various levels of the proposed framework, Algorithm III initiates the corrective action on its corresponding respondent node(s).

Algorithm 2: Threshold-based Intrusion Detection Algorithm

```
1    Div_id ← DeviceID
     Div_Data ← DeviceData,
     TD_Set ← ThresholdDataset,
     V_Status ← VulnerableStatus,
     V_Status ← inactive
     Req_id ← RequestID
2    if Req_id = active then
3        Get Div_id
4        Get Div_Data(Div − id)
5        while TD_Set do
6            Check TD_Set (Div_Data)
7            if Div_Data is within specified TD_Set Values
8                Call Algorithm I for Div_id and Div_Data
9            else
10               V_Status (Div_id) ← active
11               Call Algorithm III for Div_id and Div_Data
15           end
16       end
17   end
18   end
```

Algorithm 3: Security Insurance Algorithm

```
1    Div_id ← DeviceID
     Div_Data ← DeviceData,
     V_Status ← VulnerableStatus,
     V_Status ← active
     CA_Log ← CorrectiveActionLog
2    if V_Status = active then
3      Get Div_id
4      Get Div_Data(Div − id)
5      while CA_Log do
6        Check CA_Log (Div_Data)
7        if CA_Log (Div_Data) > Max_Value
8          Check CA_Log − Div_id (s) & Check CA_Log − Corrective Action (s) for Max_Value
           Run all the Corrective Action (s) on thier respecive Div_id (s)
9        else if CA_Log (Div_Data) < Min_Value
```

(Continued)

Algorithm 3: Continued

```
10          Check CA_Log − Div_id (s) & Check CA_Log − Corrective Action (s) for Min_Value
             Run all the Corrective Action (s) on thier respecive Div_id (s)
15        end
16      end
17    end
18    end
```

5.4 Performance Upgradation Mechanism

The performance of the proposed framework has been upgraded, employing various architectural and logical changes in the framework. The following significant changes are mainly responsible for the performance upgradation of the proposed framework:

1. *Incorporating of Edge node between IoT and Fog layer:* In most of the real-time and scalable applications, the response rate of the respondent node must be good enough to tackle real-time issues. Incorporating the edge node between IoT and Fog layer increasing the storage and computation powers of respondent nodes. It allows running intelligent algorithms i.e., Artificial Intelligence (AI) or Machine Learning (ML) based algorithms, near the respondent nodes [17]. The distance between these intelligent respondent nodes from IoT devices is larger than the Edge Nodes. Keeping the respondent nodes near the IoT devices improves the performance of the overall framework. Moreover, in other architectures that work on real-time and scalable applications, the fog and cloud node need to serve most of the requests initiated by IoT devices, affecting the framework's overall performance. By including the Edge node between IoT and Fog layer, most of the request that needs less computation power and with low latency are served by the Edge nodes. It will also improve the performance of the overall framework.
2. *Load Balanced Intelligent Algorithms:* The data and their associated methods are segregated as L-I, L-II, and L-III. This segregation is based on the level of handling the request algorithm. All L-I level methods are deployed over the respective Edge nodes. Similarly, the L-II and L-III are deployed over the fog and cloud nodes, respectively. The requests that require more storage and computational powers assigned to Edge nodes are transmitted to the L-II level, i.e., Fog nodes. Similarly, the request is only sent to the upper-level hierarchy if the current level nodes have neither any handling method available nor require computational powers [18]. This type of segregation help to balance the workload at each level, and proper utilization of resources can be achieved by making such arrangements. This improves the overall performance of the framework.
3. *Distributed Cloud Nodes:* The proposed framework recommends using Cloud Nodes in a distributed manner where each node dedicatedly performs a group of tasks. A few cloud nodes can serve requests for a specific geographical area in a comprehensive network. Similarly, other cloud nodes can serve all requests in another geographical location. Incorporating this type of distributed organization in the network helps in balancing the cloud layer load and also improves the overall network latency. As a result, the performance of the framework enhances a lot.

6 Result and Discussion

We are considering four cases to study the proposed system's performance and its comparison with other techniques. In Case-I, the combination of IoT and Cloud Computing is considered. The data acquired from IoT devices are transmitted to the central cloud node to serve all requests. In Case-II, IoT, Fog, and Cloud Computing are considered. The acquired data from IoT devices are classified by the nearest Fog nodes and transmitted to the central Cloud Node to serve all requests. In Case-III, the IoT, Edge, and Cloud Computing are considered. The data classification part is executed by the Edge node instead of the Fog nodes in the previous case. All other architectural configurations are the same as of Case-II. Finally, in Case-IV, i.e., the proposed model IoT, Edge, Fog, and Cloud Computing have been considered along with various load balancing techniques as discussed in previous sections of this study. Further, ten different ML-based algorithms have been deployed for experimentation. 100 IoT devices have been simulated for data acquisition. A dataset of 10000 request entries has been considered for a single rotation of experimentation, i.e., 100 data chunks from a single IoT device. In the subsequent sequential cycles, an additional 10000 entries have been considered. As discussed earlier, five rotations of such data chunks have been examined for each case. In all the Cases, including Case-I, II, and III, all these ten algorithms have been executed by Cloud Node only except the data classification algorithm. In Case IV, all the algorithms have been handled as specified in Table 3: "*Experimentation Environment*". Further, the detailed elaborations are presented in Fig. 4: "*Experimentation Scenario*." Moreover, in Case-IV, three cloud nodes have been deployed in a distributed manner. Ten different algorithms termed A1 to A10 are deployed for experimentation where specifically for Case IV, A1 is a data classification method, A2 is an intrusion detection and alert generation algorithm for Edge Devices, A3 to A4 and fog level intrusion detection, alert generation, and associated cloud identification algorithms, and finally A5 and A6 are intrusion detection and alert generation algorithms for cloud C1 and similarly A7 and A8 are associated with cloud C2 and A9 and A10 are associated with cloud C3 for intrusion detection and alert generation. In the absence of a particular layer, the corresponding algorithms are shifted to the next level layer available in different cases.

Table 3: Experimentation environment. Tech.-Technology Deployed, Algo.-Algorithm, A1-Classification Algorithm, A2-A10 Different ML based Algorithms, C1, C2, and C3-Cloud nodes

Layers	Case-I		Case-II		Case-III		Case-IV	
	Tech.	Algo.	Tech.	Algo.	Tech.	Algo.	Tech.	Algo.
IoT	√	X	√	√	√	X	√	X
Edge	X	X	X	X	√	A1	√	A1-A2
Fog	X	X	√	A1	X	X	√	A3-A4
Cloud	√	A1-A10	√	A2-A10	√	A2-A10	√	C1-A5,A6, C2-A7,A8, C3-A9,A10

Further, all the cases are evaluated for Data Classification, Response Rate, Intrusion Detection, and Prediction. The experimental results are drawn in Fig. 5. In classification analysis, all the cases register slight variation. Case-I took the highest average classification time of 56.64 seconds, whereas Case-IV took a minimum average classification time of 47.54 seconds. Case-IV records an average improvement of 5.9% compared to all three other cases. Continuing next to the Response Rate, The response Rate of all the first three cases is almost similar as cloud nodes have served all the requests only

in all the cases except the slight variation of the classification algorithm. Case-IV registered an average improvement of 18.56% compared to all the other three cases. Similarly, for intrusion detection, a significant average improvement of 18.45% is recorded by Case-IV. Finally, for average prediction, an improvement of 19.05% has been recorded by Case-IV.

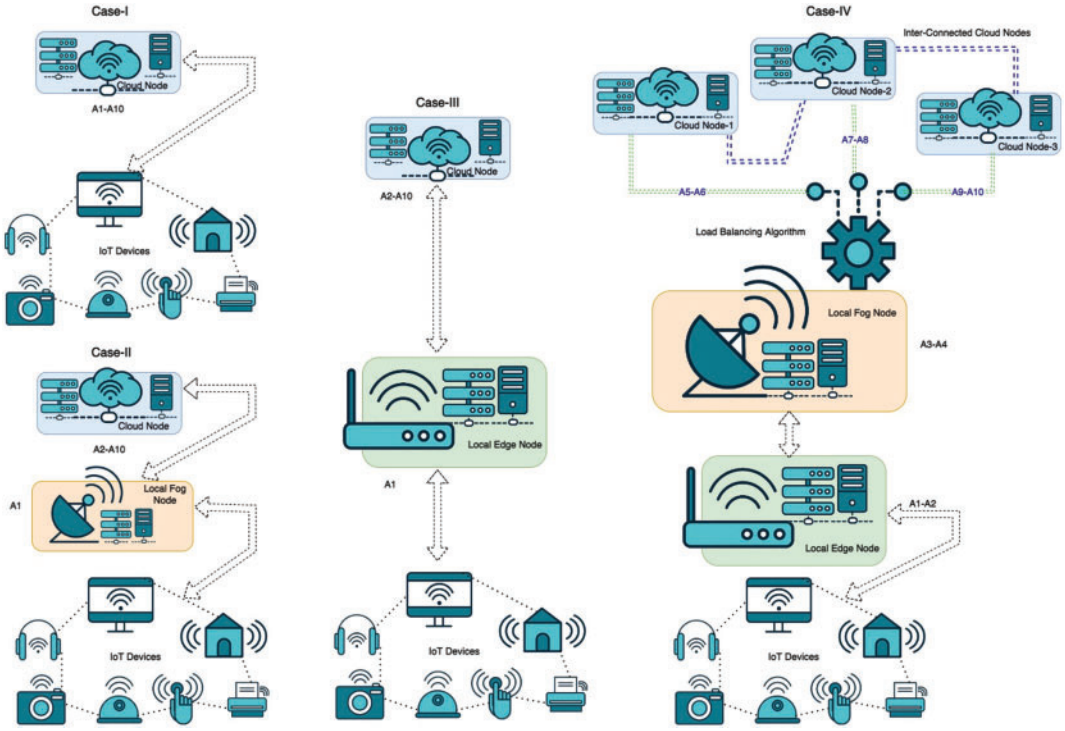

Figure 4: Experimentation scenario

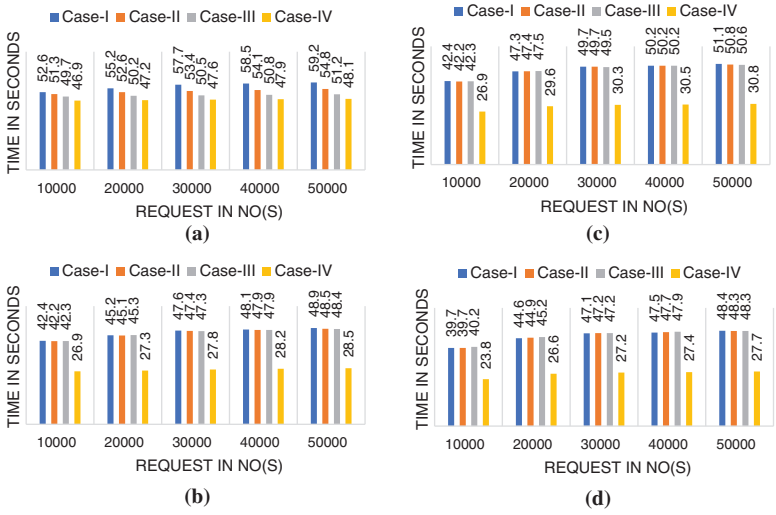

Figure 5: Performance evaluation. (a) Classification, (b) response rate, (c) intrusion detection, and (d) prediction

7 Conclusion

The involvement of state-of-the-art technologies, including IoT, Fog Computing, Edge Computing, and Cloud Computing, has been observed in all walks of life. In a decade, all these emerging technologies were used in a centralized manner for non-scalable applications. Future technology requires a real-time and scalable application. These applications require intelligent decision-making systems with reasonable response rates. This study proposes an EoT-inspired framework for real-time and scalable applications. The proposed framework integrates all the robust technologies, including IoT, Edge, Fog, and Cloud Computing. IoT devices, sensors, and actuators are deployed for real-time data acquisition in this framework. Subsequently, the local Edge nodes, Fog, and cloud nodes are employed in a distributed manner to improve the response rate of the overall framework. Threshold-based data classification and intrusion detection approaches are incorporated at various framework levels to improve the response rate and minimize the respondent nodes' distance from the IoT layer. For load balancing, algorithm segregation approaches are implemented. The threshold-based data classification and algorithm segregation for load balancing significantly improve the performance of the proposed framework. The proposed framework is implemented over the EdgeCloudSim and FogNetSim++ simulation environments. The comparison of performance evaluation of the proposed framework is executed with other technological integration, including IoT and Cloud Computing; IoT, Fog, and Cloud Computing; Edge and Cloud Computing. Performance evaluation has been implemented based on data classification, response rate, intrusion detection, and prediction. Compared to the other technology, the proposed framework's performance is examined by average improvement in respective parameters. The proposed framework recorded a 5.9% average improvement in Classification, 18.56% in Response Rate, 18.45% for intrusion detection, and 19.05% for prediction. Based on the performance results, it can be concluded that the proposed framework outreaches its other companion technological architectures for real-time and scalable applications. It can be further concluded that the decentralized approach is more suitable for real-time and scalable applications instead of the centralized approach.

Acknowledgement: The authors would like to thank the research chair of Prince Faisal for Artificial Intelligence (CPFAI) for supporting this research work.

Funding Statement: The authors would like to thank the research chair of Prince Faisal for Artificial Intelligence (CPFAI) for supporting this research work.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] M. Abunaser and A. A. Alkhatib, "Things smart home," in *Proc. of 2019 IEEE Jordan Int. Joint Conf. on Electrical Engineering and Information Technology*, Amman, Jordan, pp. 58–62, 2018.

[2] T. Banerjee and A. Sheth, "IoT quality control for data and application needs," *IEEE Intelligent Systems*, vol. 32, no. 2, pp. 68–73, 2017.

[3] M. G. R. Alam, M. S. Munir, M. Z. Uddin, M. S. Alam, T. N. Dang *et al.,* "Edge-of-things computing framework for cost-effective provisioning of healthcare data," *Journal of Parallel and Distributed Computing*, vol. 123, no. 3, pp. 54–60, 2019.

[4] A. Verma, A. Singh, E. Lughofer, X. Cheng and K. Abualsaud, "Multilayered-quality education ecosystem (MQEE): An intelligent education modal for sustainable quality education," *Journal of Computing in Higher Education*, vol. 33, no. 3, pp. 551–571, 2021.

[5] A. S. Almogren, "Intrusion detection in Edge-of-Things computing," *Journal of Parallel and Distributed Computing*, vol. 137, no. 8, pp. 259–265, 2020.

[6] S. Hamdan, M. Ayyash and S. Almajali, "Edge-computing architectures for internet of things applications: A survey," *Sensors (Switzerland)*, vol. 20, no. 22, pp. 1–52, 2020.

[7] P. Porambage, J. Okwuibe, M. Liyanage, M. Ylianttila and T. Taleb, "Survey on multi-access edge computing for internet of things realization," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 2961–2991, 2018.

[8] A. Soua and S. Tohme, "Multi-level SDN with vehicles as fog computing infrastructures: A new integrated architecture for 5G-VANETs," in *21st Conf. on Innovation, Clouds, Internet Networks, ICIN 2018*, Paris, France, pp. 1–8, 2018.

[9] T. Suganuma, T. Oide, S. Kitagami, K. Sugawara and N. Shiratori, "Multiagent-based flexible edge computing architecture for IoT," *IEEE Networks*, vol. 32, no. 1, pp. 16–23, 2018.

[10] S. M. Zhang and A. K. Sangaiah, "Reliable design for virtual network requests with location constraints in edge-of-things computing," *Eurasip Journal on Wireless Communications and Networking*, vol. 2018, no. 1, pp. 1–10, 20182018.

[11] Z. J. Sun, A. Duncan, Y. Kim and K. Zeigler, "Seeking frequent episodes in baseline data of In-Situ Decommissioning (ISD) sensor network test bed with temporal data mining tools," *Progress in Nuclear Energy*, vol. 125, no. 3212, pp. 103372, 2020.

[12] G. A. Montes and B. Goertzel, "Distributed, decentralized, and democratized artificial intelligence," *Technological Forecasting and Social Change*, vol. 141, no. 2018, pp. 354–358, 2019.

[13] K. Sadaf and J. Sultana, "Intrusion detection based on autoencoder and isolation forest in fog computing," *IEEE Access*, vol. 8, pp. 167059–167068, 2020.

[14] G. Danapal, G. A. Santos, J. P. C. L. da-Costa, B. J. G. Praciano and G. P. M. Pinheiro, "Sensor fusion of camera and LiDAR raw data for vehicle detection," in *Proc. of 2020 Workshop on Communication Networks and Power Systems (WCNPS)*, Brasilia, Brazil, pp. 1–6, 2020.

[15] S. K. Singh, S. Rathore and J. H. Park, "BlockIoTIntelligence: A Blockchain-enabled intelligent IoT architecture with artificial intelligence," *Future Generation Computer Systems*, vol. 110, no. 2, pp. 721–743, 2019.

[16] M. Anuradha, G. Mani, T. Shanthi, N. R. Nagarajan, P. Suresh *et al.,* "Intrusion detection system for big data analytics in IoT environment," *Computer Systems Science and Engineering*, vol. 43, no. 1, pp. 381–396, 2022.

[17] S. Sarkar, R. Wankar, S. N. Srirama and N. K. Suryadevara, "Serverless management of sensing systems for fog computing framework," *IEEE Sensors Journal*, vol. 20, no. 3, pp. 1564–1572, 2020.

[18] H. A. Alharbi and M. Aldossary, "Energy-Efficient Edge-Fog-Cloud architecture for IoT-based smart agriculture environment," *IEEE Access*, vol. 9, pp. 110480–110492, 2021.