Fog-Based Smart Infection Detection System (SIDS) Architecture Using Energy-Efficient Geese Optimization Load Balancing Approach (GOLBA)

Gaurav Goel * and Amit Kr Chaturvedi

Department of Computer Application, Government Engineering College, Ajmer, BTU, Bikaner, India
Email: gaurav.itindia@gmail.com (G.G.); amit0581@gmail.com (A.K.C.)
*Corresponding author

***Abstract*—Researchers are developing wearable air-borne infectious disease sensors. Since it is closer to IoT devices, the fog paradigm will boost real-time vital applications to fulfill cloud temporal limits like data throughput, response time, energy, etc. Real-time environments like intelligent healthcare monitoring suffer from inefficient load-balancing approaches for resource management at the fog layer, which reduces service quality and accelerates energy overuse. This paper proposes a fog-based Smart Infection Detection System (SIDS) architecture to control perilous infectious disease outbreaks before they spread and control pandemics such as COVID-19. An energy-efficient nature-inspired Geese Optimization Load Balancing Approach (GOLBA) is also proposed to dynamically select the closest, most active, and most resourceful computing machines to serve user requests and perform inter-cluster global and local load balancing. The proposed approach (GOLBA) outcomes are compared with the existing load balancing methods, such as the Fog Node Placement Algorithm (FNPA) and Round Robin (RR), to show its significance using the iFogSim simulator experimental setup. Analysis shows that GOLBA reduces response time by 6.6% and energy utilization by 8.9% compared to FNPA, 13.7%, and 15.1% compared to RR policy.**

***Keywords*—biosensors, load balancing, internet of things, fog computing, COVID-19, iFogSim**

I. Introduction

According to Cisco'S projections, 15.14 billion current Internet of Things (IoT) devices are estimated to be connected online. By 2025, this amount is predicted to double to 29.42 billion and will exaggerate exponentially by 2030. The IoT is fast becoming a new paradigm for passing connections in many application fields as shown in Fig. 1. These applications can be enhanced by using AI and sensing-related capability of 6G communication technology. In recent years, smart wearables, intelligent industrial and utility components, and smartphones, have grown rapidly and can detect real-time environmental data [1].

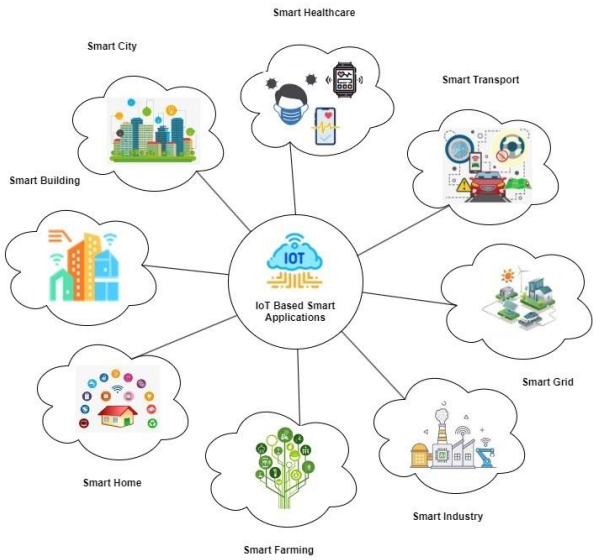

Fig. 1. Applications of IoT.

Heterogeneous IoT devices create massive volumes of data, requiring extensive processing, aggregation, and analysis to deploy innovative applications in real scenarios. Due to resource constraints, IoT devices cannot handle data, only utilizing their resources for outsourcing data processing to cloud data centers. As cloud data centers receive more requests, network congestion increases since they are centrally situated and demand more bandwidth from the leading network. The global dispersion of billions of IoT devices could exacerbate these effects for real-time applications, and leveraging outside cloud resources is unwise.

Fog computing solves these underlying IoT-cloud issues by mediating between data centers and IoT devices. Fog computing uses network edge devices to process more delay-sensitive task demands, resources, and users [2, 3]. However, fog devices' resource and energy utilization are

Manuscript received January 15, 2024; revised March 11, 2024; accepted April 22, 2024; published November 27, 2024.

challenges with fog computing. Several IoT devices seek services from the same fog device, because of which overloading may occur and negatively impact service quality. Innovative applications like competent healthcare systems inadequate resource management, and inefficient load balancing can significantly reduce fog nodes' service quality and accelerate energy overuse [4, 5].

Traditional fog computing systems overuse resources and energy due to uneven load distribution, making data collection and analysis difficult in large geographical regions. A fog-based Smart Infection Detection System (SIDS) architecture is proposed to solve these difficulties. Any air-borne infectious disease epidemic can be managed swiftly with this design, including COVID-19. SIDS aims to monitor patients' health through a face mask containing biosensors using edge and fog devices microservices. As the intelligent healthcare system is time-sensitive [6, 7], an energy-efficient nature-inspired Geese Optimization Load Balancing Approach (GOLBA) is proposed for the fog paradigm to get optimal results.

The proposed research favors low-energy processing with lesser response time and integrating more permanent elements into the system for various other potential applications to provide quality services to heterogeneous end users. The application of innovative wearable technology, also called synthetic biology in the Internet of Medical Things (IoMT), enables society to provide simple, precise infectious agent diagnosis services outside of testing laboratories. This proposed architecture can drastically enhance infection testing frequency without patient lab visits in situations like COVID-19 in a country such as India, the world's most populous nation. The summary of this paper's contributions is as mentioned below:

The proposed Fog-based Smart Infection Detection System (SIDS) architecture breaks the network into multiple planes: cloud plane, fog plane, and Smart IoT plane. This can quickly manage air-borne infectious diseases and pandemics such as COVID-19. The Geese Optimization Load Balancing Approach (GOLBA), an energy-efficient nature-inspired load-balancing technique, has been proposed for the fog paradigm. By using this method, congestion can be avoided, and network performance can be enhanced. Three performance indicators are used to evaluate the effectiveness of the GOLBA for boosting the proposed Smart Infection Detection System (SIDS), including response time, energy utilization, and cost. To demonstrate the efficacy of the GOLBA in comparison to a Fog Node Placement Algorithm (FNPA) and Round Robin (RR) methods, a wide range of simulations are run using the iFogSim toolkit. When comparing GOLBA to FNPA and RR implementation, GOLBA experimental findings demonstrate a considerably balanced load distribution, a drop in energy utilization, and a reduced response time.

The rest of this study is compiled as follows: A current literature study is explained in Section II. Section III explains the proposed fog-based architecture, resource allocation algorithm, load-balancing approach, and simulation setting. The findings and discussion of results after simulations are shown in Section IV. The conclusion with recommendations for further research is given in Section V.

II. Related Works

Yasmeen *et al.* [8] recommended a Particle Swarm Optimization (PSO) load balancer with simulated annealing for smart buildings, considering Global best (Gbest) underpins PSO functionality. However, the initial allocation of Local best (Lbest) causes an issue for the Simulated Annealing Algorithm. The authors concluded that PSO's performance can be enhanced by picking the Gbest using simulated annealing after each iteration, which is the updated Gbest. The authors avoids discussing fog nodes' response time-energy trade-off. A round-robin priority-based load balancer was presented by Pereira *et al.* [9]. In the proposed load balancer, network administrators chooses fog nodes for load balancing based on job priority queues and observe high-priority processes finish faster because low-priority jobs are queued up last.

Beraldi *et al.* [10] proposed sequential forwarding, adaptive forwarding with self-tuning ability load balancing for smart cities, considering node heterogeneity and uneven load distribution. It limits node transmissions by altering a threshold using a memoryless assessment of their present state. The processing queue fills up after forwarding a task as feasible, and the job is abandoned. Fatima *et al.* [11] introduced an intelligent building proximity dynamic service broker policy. The method chooses fog because it can handle load depending on virtual machine allocation and decrease network latency. However, cost and response time are required to be balanced. Chekired *et al.* [12] proposed the intelligent electric car priority queuing model. The goals were to minimize delay and provide geo-aware services for the intelligent grid model while allowing electric car customers optimal charging and unplugging planning.

Zahid *et al.* [13] proposed load balancing for intelligent grids using the hill-climbing approach. It finds available virtual machines using mathematical optimization and randomness. Requests are sent to the best Virtual Machine (VM). However, the cost-response time trade-off remains unresolved. Butt *et al.* [14] developed Binary PSO, a genetic algorithm for intelligent cities. The authors proposed that meta-heuristic techniques allows to swiftly reach global solutions while making trade-offs, unlike heuristic algorithms, which can become stuck in local optima. Improved PSO with levy walk was developed by Khan *et al.* [15] for intelligent grid load balancing. The author noticed heuristic algorithms often get stuck in local optima and summarized that meta heuristic algorithms assist in swiftly finding global answers with trade-offs.

Hussein and Mousa [16] developed a meta-heuristic load balancer using Ant Colony Optimization (ACO) and PSO to improve end-user application Quality of Service (QoS). The authors suggested maintaining fog node task stability is crucial and observed ACO and PSO are superior to greedy search. The authors determined that multi-objective improvement must include cost and power utilization for better results. Divya [17] suggested an

intelligent camera SDN, deep reinforcement (Q Learning Algorithm). The reinforcement learning method used experience and dynamic network features to pick intelligent behaviors. Ali *et al.* [18] proposed a modified genetic approach resource allocation method for solving discrete optimization issues of assigning a computational node that cannot be assigned using a continuous space. The authors did not allow independent service composition reconfiguration at the network edge.

Hameed *et al.* [19] introduced vehicular fog for job execution in autonomous cars. Here, clustering may pick a cluster master based on the fog movement factor. Another method proposed was capacity-based load balancing to alleviate network congestion. The authors did not focus on sophisticated, dynamic self-learning approaches to boost energy utilization and service quality. Greedy techniques move the burden from the overloaded fog node to the one having the lowest load and let the underloaded computing nodes share the extra load. A coalition game-based algorithm for load balancing by Yan *et al.* [20] maximizes reward. Asghar *et al.* [21] proposed an improved load-balancing method for health monitoring systems. The authors presented that dataflow delays for IoT devices include load-related computation latency and traffic load-related communication latency. Thus, base station and fog node compute loads were considered during load balancing. The authors should have discuss the fog node latency, cost, and energy trade-off for optimal results.

Wan *et al.* [22] projected an energy aware multi-agent system for an intelligent factory (Candy Packing) to evaluate energy utilization and equipment workload. Fog nodes run an innovative factory energy model. The authors presented load balancing optimization using a multi-agent system and enhanced PSO to aid work scheduling. Alzeyadi *et al.* [23] extended the PSO evolutionary resource allocation algorithm for intelligent factory robots (Drug Packing), where fog applications consider the threshold to help intelligent industrial robots manage their workload. Server overload and underutilization reduce network performance and increase power use. The authors summarised that model-based intelligent learning is needed to replicate bias-free findings.

Li *et al.* [24] presented cloud-fog cooperation scheduling with task offloading. The authors showed that low latency frequently uses more energy. Energy optimization is achieved by trading off traffic load on each tier using NL programming. However, heterogeneous fog testing is required. Kaur and Aron [25] proposed an energy-efficient method in which end users send queries to the fog layer, which utilizes a load balancer to evaluate virtual machine capacity and performance. Overloaded VMs strain underloaded ones. Bala and Chishti [26] presented EEG Tractor Beam, a proximity and Cluster Algorithm (CA) for online games. The proposed approach places application modules on the nearest fog device to decrease transmission delays. CA consolidates numerous modules onto a single device to save network bandwidth utilization.

DRAM was suggested by Xu *et al.* [27] and used dynamic service mobility and immobile resource allocation to balance loads. However, the authors did not consider the fog node load movement's pros and cons. A method for locating and keeping an eye on clusters during the pandemic's first, second, and third phases was offered by Herath *et al.* [28]. The framework has been built using technologies related to fog computing, smart sensors, and IoT-based approaches. The component was inefficient in boosting the testing frequency required to stop the spread of the COVID-19 pandemic.

Goel and Chaturvedi [29, 30] showed that real-world services' QoS needs are significant when fog nodes are less than end-user task-demanding services. Second, Fog Nodes (FNs) are constantly congested, which lowers efficiency and QoS for all FNs. The authors concluded that multi-objective optimization methodologies for resource allocation and load balancing can improve fog node clusters with superior service quality. Advanced hybrid meta-heuristic algorithms and learning-based adaptive and intelligent solutions are needed for optimal results in real applications. The existing load-balancing methods used for smart scenarios with key QoS parameters are listed in Table I.

After a rigorous literature review, the following recurring gaps are identified in existing research works mentioned in Table I. Previous studies [8–15] need to optimally handle the required trade-off between response time and energy consumption of fog nodes. The existing study [16–21] does not simultaneously include multi-objective optimization such as response time, power consumption, and cost. The different approaches [22–27] focus on something other than self-learning-based dynamic and intelligent techniques to improve the quality of service and energy consumption. Only some academics implemented their fog-based architecture for intelligent healthcare systems. Most research only assessed their techniques against delay or response time, ignoring trade-offs between other performance indicators like energy, cost, etc.

Looking at the recurring challenges that researchers are addressing across different approaches. It is observed that smart healthcare monitoring systems have extensive requirements of improved load-balancing approaches for effectively handling the exponential heterogeneous IoT load on the fog layer. There is an urge for optimal resource and energy utilization, and for continually providing quality services in the upcoming future demands. The fog-based Smart Infection Detection System (SIDS) architecture is proposed along with energy-efficient Geese Optimization Load Balancing Approach (GOLBA). It covered the significant requirement of multi-objective optimization and effectively handled the trade-off between response time and energy consumption for optimal results.

TABLE I. SUMMARY OF EXISTING LOAD BALANCING METHODS WITH PERFORMANCE METRICS

References	Scenario	Existing Approaches	Tool Used	RT	ET	DL	EG	RU	CT
[8]	Smart Building	PSO with Simulated Annealing	Cloud Analyst	√					
[9]	Generic Scenario	Priority Round Robin Load Balancer	Not Mentioned	√					
[10]	Smart City	Sequential and Adaptive Forwarding	Not Mentioned	√					
[11]	Smart Buildings	Service Broker Dynamic Service Proximity	Cloud Analyst	√					√
[12]	Electric Vehicles	Priority Queuing Model	NS-2	√			√		
[13]	Smart Grid	Hill Climbing	Cloud Analyst	√	√				√
[14]	Smart City	Binary PSO Genetic Algorithm	Cloud Analyst	√	√				√
[15]	Smart Grid	Improved PSO with Levy Walk (IPSOLW)	Cloud Analyst	√	√				√
[16]	Generic Scenario	Ant Colony and Particle Swarm Optimization	Matlab	√					√
[17]	Smart Cameras	SDN, Deep Reinforcement Learning Algorithm	Not Mentioned		√				
[18]	Generic Scenario	Non-dominated Sorting Genetic Algorithm II	Matlab		√				√
[19]	Auto Driven Vehicles	Vehicular Fog Computing	NS-2			√			
[20]	Generic Scenario	Coalition Based Greedy Algorithm	NS-3			√	√		
[21]	Health Monitoring	Fog Node Placement Load Balancing	iFogSim			√			
[22]	Smart Factory	Energy Aware LBS, IPSO	Not Mentioned				√		
[23]	Smart Factory	Extended ELBS (PSO Evolutionary Algorithm)	iFogSim			√	√		
[24]	Generic Scenario	Cloud-Fog Cooperation Scheduling	Not Mentioned			√	√		
[25]	Generic Scenario	Energy Aware LBA	iFogSim			√	√		√
[26]	Online Games	Proximity and Cluster Algorithm	iFogSim				√		√
[27]	Generic Scenario	Dynamic Resource Allocation Method	CloudSim					√	

III. EXPERIMENT & METHOD

A. Proposed Fog Based Layered Architecture

This section describes a three-tier fog-based Smart Infection Detection System (SIDS) architecture to control pandemics like COVID-19 and avoid further epidemics quickly. The proposed fog-based SIDS architecture uses face masks equipped with biosensors [6] to diagnose air-borne infectious diseases such as COVID-19 using a smartphone and fog computing microservices. The recommended architecture has three tiers. The first tier of architecture uses biosensors on the patient's facemask to detect and transmit COVID-19 vital indicators. IoT devices are linked to specific server stations, but their coverage zones may overlap, increasing compute node availability under heavy traffic loads [31].

All IoT devices are connected with server stations with heterogeneous fog gateways at the intermediate smart healthcare fog layer. The first-tier IoT devices provide sensor-perceived data to nearby fog gateways. A second-tier fog gateway linked with the cluster controller node monitors each patient's COVID-19 status. The vital signs are analyzed to diagnose COVID-19 and identify criticality based on the patient's history at the second tier. A top-layer proxy server sends observations to the third tier's cloud data centers for storage. Simultaneously, fog gateways also send patients' COVID-19 test results to patients' smartphones. The proposed three-tier fog-based Smart Infection Detection System (SIDS) layered architecture is presented in Fig. 2.

I Tier: Smart IoT Plane covers the perception layer, which comprises biosensors attached to patients via face masks, to sense and pass the vital signs of air-borne infectious diseases such as COVID-19 to fog gateways. The sensor-collected data stream is transmitted to the fog plane through an edge device like a smartphone, and results are displayed on actuators. Patients are notified of their requests during the filtering-processing phase within the IoT layer. The patient receives results and preventive measures through actuators if the fog gateway sample test is positive. The rapid sample testing-processing phase is conducted on the fog layer, avoiding visiting laboratories physically and supporting real-time updates and timely results, which in turn assist in fast spread control of air-borne infectious diseases such as COVID-19 before it becomes a pandemic. Computation at the fog layer ensures the quality of services to even those end users with resource-constrained IoT devices so that this system can benefit the community as a whole.

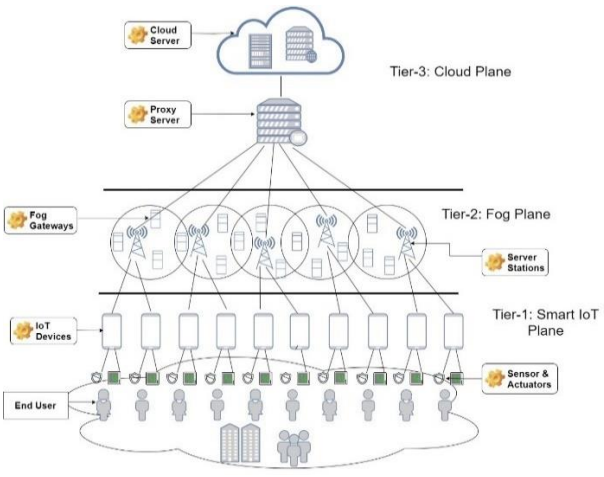

Fig. 2. Fog-based Smart Infection Detection System (SIDS) layered architecture.

II Tier: The fog plane covers the intermediate layer, which includes a cluster of fog gateways co-located with the cluster master node/server station to check the COVID-19 status of patients regularly. The fog gateway receives patients' task-processing requests. It sends them to a cluster controller node, which allocates tasks to the closest,

active, and most resourceful fog gateway based on quality-of-service parameters using the proposed multi-objective Optimal Nearest Fog Node Search (ONFNS) resource allocation algorithm. To connect, every network item shared a beaconing message between IoT devices, fog gateways, and cluster controller nodes. This message includes IoT devices, fog gateway and cluster controller node identifiers, geographical positions, and time stamps. Cluster controller nodes use beaconing messages to assess distance from fog gateways within their communication range and form clusters. In overloaded states, fog gateways can adapt their computing burden by adjusting IoT device affiliations between server stations and considering overlapping coverage zones, affecting fog gateway computing and traffic loads.

III Tier: Cloud plane covers proxy servers and the cloud to connect with data centers for long-term data storage, analysis of the volume of data, and providing feedback on patients' air-borne infectious diseases such as COVID-19 status in specific situations for future quick assistance. It also assists in computation during heavy traffic load on server stations or overloading of fog nodes to avoid disasters.

B. Proposed Resource Allocation Algorithm

Studies have shown that geese can teach people to command a flock in the skies. The flight range of geese in a perfect V formation is 71% greater than when flying alone. Analogously, the proposed system uses a pod of clusters to boost the IoT-Fog architecture's processing capabilities in intelligent healthcare design. This cluster shares computing and storage via many fog gateways connected to server stations. After breaking from formation, a goose realizes how hard it is to fly alone and tries to rejoin. The configuration improves a flock's visibility and power while making flying simpler for birds.

Similarly, when a cluster controller node is overburdened with task requests, the proposed system advises transferring some of the load to the next controller node. This may boost the next cluster's compute demand and decrease the previous cluster controller node's traffic. After processing all queue requests, the prior cluster controller node prepares to process new task requests.

The aim of the proposed multi-objective Optimal Nearest Fog Node Search (ONFNS) Resource Allocation Algorithm, as shown in Algorithm 1, is to find the fog gateway with the fewest distance using Algorithm 1, with adequate resources to handle the job request using Algorithm 2 and improve the energy efficiency with geese optimization Load Balancing Approach using Algorithm 3. The proposed approach (ONFNS) is also explained using the flowchart at the end of this section in Fig. 3.

The fog gateway gets end device requests within its geo coverage zone. The distant fog node transmits the task request to the cluster controller node, which allocates jobs or forwards the task request to the appropriate gateway within the cluster or neighboring cluster controller node based on their capabilities using Algorithm 1. The proposed algorithm uses the Select In-range Cluster Nodes function using Algorithm 1 to generate a route map of IoT devices and fog nodes. After that, the route chart is sorted in ascending order to select the minimum distance fog gateway for computation. Besides the shortest distance, the algorithm verifies available resources using the Compute Resourceful Cluster Node function using Algorithm 2. In addition, the nature-inspired geese optimization load balancing approach, as shown in Algorithm 3, dynamically chooses the new cluster controller node while managing many task processing at a specific time, considering fog nodes' average delay and energy utilization. Calculating fog node traffic load, energy loss, and communication delay helps to pick the efficient cluster controller node with the lowest average latency and minimum energy utilization.

Algorithm 1: Optimal Nearest Fog Node Search (ONFNS) Algorithm

Input: Group of fog gateways N, Group of end devices D
Output: Mapping of fog node & IoT Device

```
Optimal_Nearest_fog_node_search (Edevice[], Fnode[])
Initialize ClustHead[Fnode], ClustMap[Edevice], RouteLink[Edevice][Fnode], Edis[], j=0, k=0
RouteLink[][] = Select_inrange_cluster_nodes (Edevice[], Fnode[])
for each Edevice_i Ɛ Edevice do //i=0 to Edevice.Count
Edis[] = RouteLink[i][j]
Sort(Edis) //In ascending order
for each Fnode_j Ɛ Fnode do //j=0 to Fnode.Count
if(Edis[k] == RouteLink[i][j]) then
Flag = Compute_resourceful_cluster_node (i,j)
if (Flag =1) then
ClustMap[i] = j //Assign fog node
else
k=k+1 //Check next optimal fog node
Re-initialize j=0
Close if
else
j=j+1
close if
close for
i=i+1
close for
Save to stable storage (ClustMap)
ClusterHead[] = Geese_optimization_load_balancing (Edevice[], Fnode[])
Close function
```

Algorithm 2: Compute Resourceful Cluster Node

Input: Group of Fog gateways N, Set of end devices D
Output: Integer (Flag)

```
Compute_resourceful_cluster_node (Edevice e, Fnode n)
if (n.cpu >= e.cpu && n.ram >= e.ram &&n.network >= e.network &&n.energy >= e.energy) then
 n.cpu = n.cpu-e.cpu
 n.ram = n.ram – e.ram
 n.network = n.network – e. network
 n.energy = n.energy – e.energy
 Flag = 1
else
 Flag = 0
close if
Close function
```

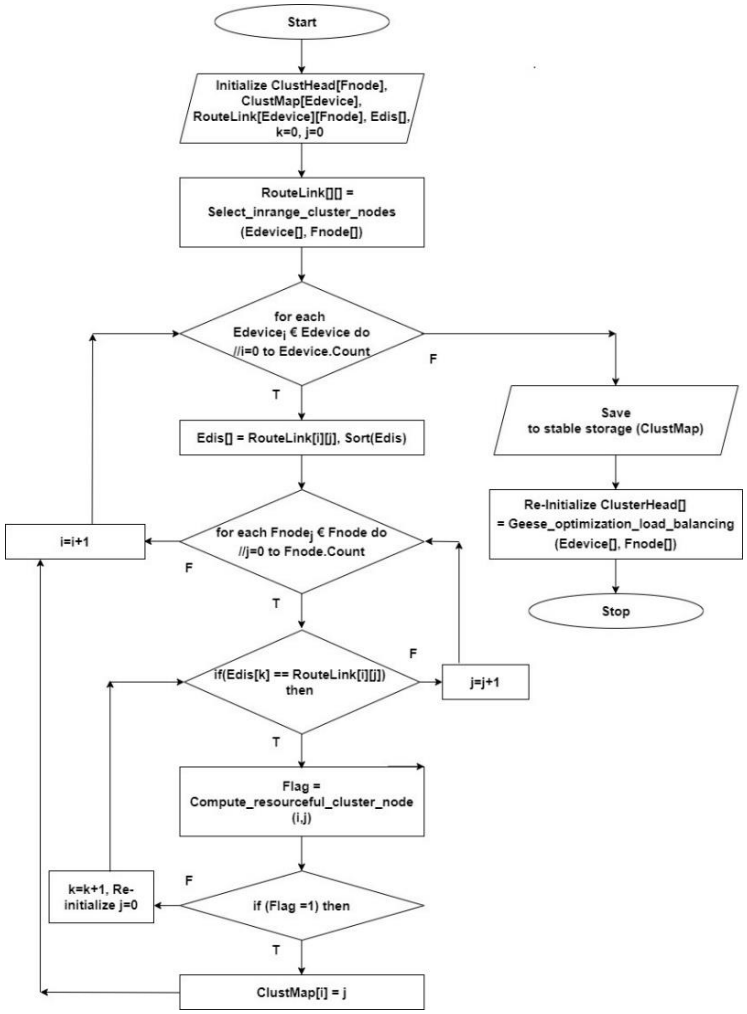

Fig. 3. Flowchart for Proposed Approach (ONFNS).

The concept of Algorithm 3 is taken from the life of a goose. When geese fly together, lead geese in the formation have high "Drag". When it tires, the lead goose returns to the formation and is replaced. The lead goose immediately feels the lifting force from the bird before it. Likewise, the proposed approach dynamically picks the new cluster controller node within a cluster, considering the average delay and energy utilization of fog gateways using Algorithm 3 while managing many task requests at a set time. This helps to retain QoS when the previous cluster controller node is congested, leading to high energy loss. Besides, Geese may honk to cheer each other. Honking is also thought to tell other geese where they are in the formation.

In the same way, every network item sends a beaconing message to connect the fog gateways and cluster controller nodes. This message includes fog gateway and cluster controller node identifiers, geographical positions, and time stamps. Cluster controller nodes use beaconing messages to assess distance from fog gateways within their communication range. These linked fog nodes lead to the reformation of clusters. The proposed approach manages and improves SIDS, an intelligent healthcare monitoring system task distribution, resource utilization, response time, energy efficiency, and overall service quality.

Algorithm 3: Geese Optimization Load Balancing Approach (GOLBA)

Input: Set of Cluster nodes N, Set of Edge devices D
Output: Dynamic selection of cluster master node

1. Geese_optimization_load_balancing (Edevice[], Fnode[])
2. Initialize ClustHead, Ndelay[Fnode][Edevice], Nenergy[Fnode][Edevice], Navgdelay[ ], Navgenergy[]
3. for n Є N do //n=0 to fognode.count
4. for d Є D do //d=0 to edgedevice.count
5. if d Є cover_range(n) then
6. Calculate Dcm(d) //Communication delay
7. Calculate Ecm(d) //Communication energy
8. close if
9. Ndelay[n][d] = Dcm(d)
10. Nenergy[n][d] = Ecm(d)
11. close for
12. Navgdelay[n] = Average(Ndelay)
13. Navgenergy[n] = Average(Nenergy)
14. close for
15. Select ClustHead←min(Navgdelay, Navgenergy)
16. Return ClustHead

The proposed SIDS can boost testing frequency without forcing patients to visit the laboratories during a pandemic. It meets the QoS needs of delay-sensitive IoT-based innovative healthcare applications by migrating huge

tasks. It requests data from IoT sensor nodes to fog nodes, which are closest, more active, and resourceful, using the ONFNS algorithm. When the intelligent healthcare system is implemented widely, the server station balances the load inside a cluster using a proposed algorithm with optimal response time and energy utilization.

C. Experimental Setup

1) Simulation setup for fog-based SIDS

Sensors that detect air-borne infectious diseases such as COVID-19 test parameters communicate data to fog gateways through smartphones. The computing fog gateways process and analyze data to evaluate the patient's COVID-19 diagnostic status and determine if they are critical or normal based on the patient's prior history. Fog gateways provide patient feedback to their smartphones and store it in the cloud. Fog gateways and cloud servers communicate via proxy servers. Our technique was simulated and evaluated using iFogSim [32], an open-source toolbox.

iFogSim uses Cloudsim's API to manage its discrete event-based simulation in Java. This high-performance open-source fog computing, edge computing, and IoT toolkit models and simulates their networks. iFogSim combines resource management approaches that can be customized for study. The following iFogSim classes are needed to simulate the fog network: Fog device, Sensor, Actuator, Tuple, Application, Monitoring edge, and Resource management service for architecture layer specifications and virtual machine simulation setup.

The current fog computing application simulator is iFogSim, which is used to construct fog models with IoT integration for experimental validation. iFogSim simulation environment is required to define CPU length, RAM, bandwidth, and other characteristics while designing fog devices. The proposed Resource Allocation Algorithm (ONFNS) with a Load Balancing Approach (GOLBA) has included cost, energy utilization, and response time as performance parameters. The iFogSim network configuration options for the experimental setup is shown in Table II.

TABLE II. NETWORK CONFIGURATION FOR FOG BASED ARCHITECTURE

Param. / Comp. Device	CPU (mips)	RAM (mb)	UP-BW (mbps)	DW-BW (mbps)	Latency (ms)	Rate (Cost/Mips)
Cloud	44,800	40,000	10,000	10,000	100	0.01
Proxy	12,800	8,000	10,000	10,000	10	0.04
Fog Gateway	12,800	8,000	10,000	10,000	5	0.08
IoT Device	2,800	4,000	10,000	10,000	10	0.16

Devices in a computer network, however, come in various configurations. At the root level, a cloud server acts as the parent. The fog gateways can communicate with the cloud server using the Level 1 proxy server. Level 2 computing gateways are situated closer to the end user to offer computing and storage resources more often. The third tier of IoT devices is sensor and actuator-based. The fog computing-based solution's evaluation architecture, using iFogSim, is depicted in Fig. 4. The HP laptop (Inteli3, 2.60 GHz processor, 256 GB SSD drive) was used for the simulations.

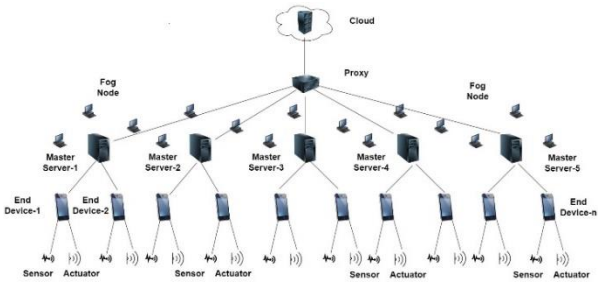

Fig. 4. Fog-based smart healthcare system topology in iFogSim.

The proposed design was simulated 500 times with several topologies using iFogSim. The architecture has been altered by adding additional IoT/End Devices (ED) and Fog Nodes (FN). Two fog nodes were initially linked to server stations, with ten IoT devices linked within the cluster. IoT devices and fog nodes were added afterward to broaden the topology (Topol). Simulation fog experimental setups are shown in Tables III and IV, which are examined based on response time, energy utilization, and cost, and readings are recorded for each simulation to evaluate the final results.

TABLE III. SIMULATION SETUP CORRESPONDING TO VARYING IOT DEVICES

Topology	Cloud	Proxy	Fog Server	Fog Node	IoT Device
Topol-1	1	1	5	4	100
Topol-2	1	1	5	4	200
Topol-3	1	1	5	4	400
Topol-4	1	1	5	4	800
Topol-5	1	1	5	4	1,000

The simulation setup with different topologies corresponding to a fixed number of fog nodes and a varying number of IoT/End devices is presented in Table III.

The simulation setup with different topologies corresponding to varying fog nodes and fixed number of IoT/End devices is presented in Table IV.

TABLE IV. SIMULATION SETUP CORRESPONDING TO VARYING FOG NODES

Topology	Cloud	Proxy	Fog Server	Fog Node	IoT Device
Topol-1	1	1	5	2	400
Topol-2	1	1	5	4	400
Topol-3	1	1	5	6	400
Topol-4	1	1	5	8	400
Topol-5	1	1	5	10	400

2) Performance evaluation parameters

As performance assessment factors, this paper included response time, energy utilization, and computing cost. Below is a brief explanation of each parameter:

Response Time: Response time represents the sum of all delays of any kind that occur during the processing of a biomedical request. Propagation and processing delays are included. It is computed in milli second (ms). With the use of Eq. (1) [16], the response times of the proposed and implemented techniques have been determined.

$$Rtime_{ij} = Delay_{ij}^{cm} + Ptime_{ij} \quad (1)$$

It's evident from Eq. (1) that the response time is the total of the IoT-Fog transmission delay and processing duration on the fog gateways.

Energy Utilization: The overall energy loss by the IoT devices and fog resources to do their assigned tasks is known as energy utilization. It is computed in Joules (J). With the aid of Eqs. (2) and (3) [33], the energy utilization of the proposed techniques presented as follows:

$$Energy_{Usage} = \sum_{i=1}^{n} E_{Sense\,(i)} + E_{Trans\,(i)} + E_{Proc\,(i)} \quad (2)$$

$$Energy_{Total} = E_{Active\,Node} + E_{Idle\,Node} \quad (3)$$

Eq. (2) demonstrates that the overall energy utilization is the sum of the energy used for each task's sensing, transmission, and execution. However, Eq. (3) presents that the total energy utilized by active and idle fog nodes equals the total energy used.

Computational Cost: The overall cost of resources when completing submitted tasks is known as the computational cost. It is computed in Dollars. Using Eq. (4) [33], the computational costs of proposed and implemented techniques have been determined.

$$Cost_{Comp} = \sum_{i=1}^{n} MIPSHost_i \; X \; PTHost_i \; X \; CostHost_i \quad (4)$$

The cost of computing may be understood from Eq. 4, as the total of the MIPS costs for each agent, the processing time spent by each agent, and the computational costs of each agent.

IV. Result and Discussion

This section gives the results of the performance evaluation of the GOLBA in comparison to the FNPA and RR scheme. The outcomes for the performance parameters such as response time, energy used, and cost incurred are presented in Table V. Comparing GOLBA to FNPA and the RR scheme in terms of response time and energy usage, the simulation results show that GOLBA minimizes both the parameters and produce optimal results.

It is observed that during simulation, when load distribution is uneven, some fog nodes are overwhelmed while others are idle, and long fog node processing times cause network congestion. Due to network congestion, IoT device response times are longer and may exceed job deadlines. GOLBA, an efficient load-balancing approach, helps distribute the load across idle fog nodes to minimize network congestion and improve smart healthcare applications' response time and energy utilization.

Our proposed approach in fog based implementation is an effective option for infection detection systems, as demonstrated by the experimental findings for the performance metrics of response time and energy usage. In light of these results, we can see that fog computing could work in settings where processing data quickly is crucial.

The simulation results of GOLBA with fixed number of fog nodes and growing IoT devices are calculated using the ifogsim toolkit and are presented in Table V. The response time is 896.66 ms, the energy used is 167345.10 J, and the cost incurred is 1014.65 dollars for the simulation setup Topol-1 with 100 IoT devices and 4 fog nodes, as shown in Table III. Similarly, the response time is 1564.96 ms, the energy used is 169398.26 J, and the cost incurred is 2183.49 dollars for the Topol-2 with 200 IoT devices and 4 fog nodes. The response time is 2512.28 ms, the energy used is 170129.73 J, and the cost incurred is 3782.30 dollars for the Topol-3 with 400 IoT devices and 4 fog nodes. The response time is 6388.13 ms, the energy used is 171671.47 J, and the cost incurred is 8260.83 dollars for the Topol-4 with 800 IoT devices and 4 fog nodes. The response time is 11687.41 ms, the energy used is 171935.46 J, and the cost incurred is 9134.17 dollars for the Topol-5 with 1000 IoT devices and 4 fog nodes.

TABLE V. Simulation Results of GOLBA for Varying IoT Devices

Setup	Topol-1	Topol-2	Topol-3	Topol-4	Topol-5
RT (ms)	896.66	1,564.96	2,512.28	6,388.13	11,687.41
EU (J)	167,345.10	169,398.26	170,129.73	171,671.47	171,935.46
Cost ($)	1,014.65	2,183.49	3,782.30	8,260.83	9,134.17

The analysis of the results in Table V suggests that the response time and cost incurred rise with the increase in the number of IoT devices for a fixed number of fog nodes associated with the cluster controller node because of increased communicational and computational latency, respectively. However, energy usage is increased slightly due to efficient load balancing of task requests among the fog nodes coming from IoT devices. Using the proposed approach, we can improve the result by increasing the number of fog nodes per server within a cluster with growing numbers of IoT/End devices.

The results of the proposed policy are compared with existing policy in terms of response time are plotted in Fig. 5, which reflects that the proposed strategy, i.e., GOLBA, has 6.6% less response time as compared to the FNPA and has 13.7% less response time as compared to the RR scheme. Similarly, the results of the proposed policy are compared with existing policy in terms of energy used are plotted in Fig. 6, which reflects that the proposed strategy, i.e., GOLBA, consumed 8.9% less energy as compared to the FNPA and consumed 15.1% less energy as compared to the RR scheme. Besides, the results of the proposed policy are compared with existing

policy in terms of cost incurred are plotted in Fig. 7, which reflects that the proposed strategy, i.e., GOLBA, incurred 7.7% more cost as compared to the FNPA and incurred 13.7% less cost as compared to the RR scheme.

The comparative outcomes of the GOLBA method with FNPA and RR policy concerning response time are presented in Fig. 5. The GOLBA simulation findings indicate that the proposed approach improved by 6.6% response time as compared to FNPA and 13.7% response time as compared to RR policy for a given experimental setup, having four fog nodes connected to cluster controller node and 100 to 1,000 variable end devices.

The comparative outcomes of the proposed approach with FNPA and RR policy concerning energy utilization are presented in Fig. 6. The GOLBA simulation findings indicate that the proposed approach improved by 8.9% energy utilization compared to FNPA and by 15.1% energy utilization compared to RR policy for the given experimental setup, having four fog nodes connected to cluster controller node and 100 to 1000 variable end devices.

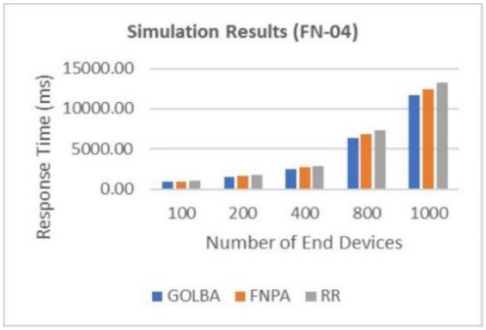

Fig. 5. Comparative results based on response time (FN-04).

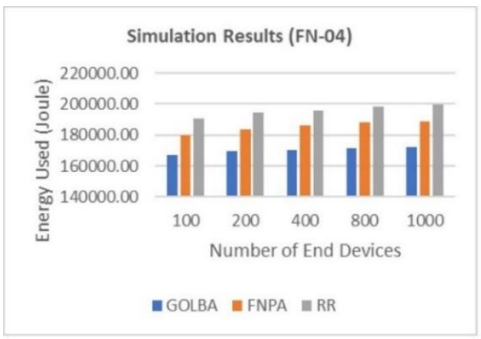

Fig. 6. Comparative results based on energy utilization (FN-04).

The comparative results of the proposed approach, with FNPA and RR policy concerning cost are presented in Fig. 7. The GOLBA simulation findings indicate that the proposed approach costs 7.7% more than FNPA, while 13.7% less cost compared to RR policy for a given experimental setup, having four fog nodes connected to cluster controller node and 100 to 1,000 variable end devices.

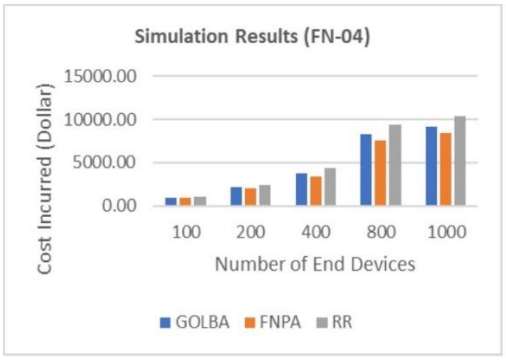

Fig. 7. Comparative results based on cost (FN-04).

The simulation results of GOLBA with growing number of fog nodes and fixed number of IoT devices are calculated using the iFogSim toolkit and are presented in Table VI. The response time is 2864.33 ms, the energy used is 172181.65 J, and the cost incurred is 3805.61 dollars for the simulation setup Topol-1 with two fog nodes & 400 IoT devices, as shown in Table IV. Similarly, the response time is 2512.28 ms, the energy used is 170129.73 J, and the cost incurred is 3782.30 dollars for the Topol-2 with four fog nodes and 400 IoT devices. The response time is 2025.86 ms, the energy used is 167084.83 J, and the cost incurred is 3766.75 dollars for the Topol-3 with six fog nodes and 400 IoT devices. The response time is 2395.00 ms, the energy used is 167067.94 J, and the cost incurred is 3785.12 dollars for the Topol-4 with eight fog nodes and 400 IoT devices. The response time is 2686.05 ms, the energy used is 167062.59 J, and the cost incurred is 3809.01 dollars for the Topol-5 with ten fog nodes and 400 IoT devices.

The analysis of the results presented in Table VI suggests that the response time and cost incurred drop with the increase in the number of fog nodes associated with cluster controller node up to a specific level for a fixed number of IoT devices. After that, both parameter values start rising due to increased communicational and computational latency. However, energy usage gradually decreases due to efficient load balancing of task requests among the fog nodes coming from IoT/End devices.

TABLE VI. SIMULATION RESULTS OF GOLBA FOR VARYING FOG NODES

Setup	Topol-1	Topol-2	Topol-3	Topol-4	Topol-5
RT (ms)	2,864.33	2,512.28	2,025.86	2,395.00	2,686.05
EU (J)	172,181.65	170,129.73	167,084.83	167,067.94	167,062.59
Cost ($)	3,805.61	3,782.30	3,766.75	3,785.12	3,809.01

The results of the proposed policy are compared within different topologies of simulation setups. The response time is plotted in Fig. 8, which reflects that the response time decreased gradually up to a specific limit of increase number of fog nodes within a cluster; after that, the response time rises due to increased communication latency. Similarly, the energy used is plotted in Fig. 9, which reflects that energy usage decreased gradually for

the added number of fog nodes within a cluster, after that, the energy usage become stable due to the availability of more active fog nodes for task request allocation. The cost incurred is plotted in Fig. 10, which reflects that the cost decreased gradually up to a specific limit of increase in fog nodes within a cluster; after that, the cost rises due to increased computational latency.

The simulation results of GOLBA concerning response time are presented in Fig. 8. iFogSim simulation results show that the response time decreased gradually up to Topol-3; after that, due to increased communication latency, the response time rises for a given experimental setup, having 2 to 10 variable fog nodes connected to cluster controller node and constant 400 end devices.

The simulation results of GOLBA concerning energy utilization are presented in Fig. 9. Simulation results show that the energy utilization decreased gradually for the experimental setup, which had 2 to 10 variable fog nodes connected to the cluster controller node and 400 end devices.

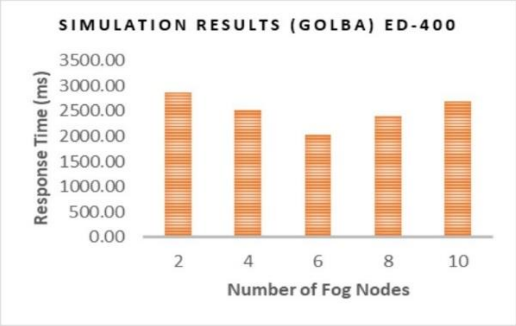

Fig. 8. Relative results based on response time (ED-400).

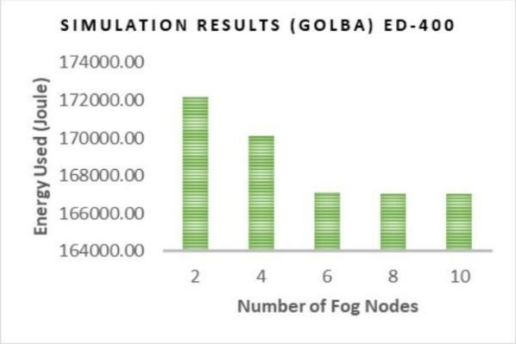

Fig. 9. Relative results based on energy utilization (ED-400).

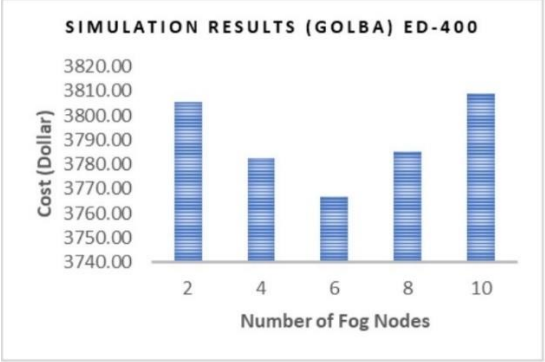

Fig. 10. Relative results based on cost (ED-400).

The simulation results of GOLBA concerning the cost presented in Fig. 10. Simulation results show that the cost decreased gradually up to Topol-3; after that due to increased computational latency, the cost rises for the given experimental setup, having 2 to 10 variable fog nodes connected to cluster controller node and constant 400 end devices.

In summary, it is observed during the simulation that the significant response time and energy savings come with a price cost, as the cost of computation is higher on fog nodes and with the proposed scheme most of the computation will now be possible on fog gateways efficiently. This paper presents two key observations; first, the proposed fog-based SIDS is quite favorable for the delay-sensitive intelligent healthcare monitoring system. Second, the results favor the GOLBA approach, considering performance compared to the FNPA and RR, which are existing load-balancing methods.

V. Conclusion and Future Work

Cloud computing architecture is employed in most smart monitoring systems. However, the massive adoption of delay-sensitive apps is slowed by the inability of the cloud to handle high data volumes in real-time. The fog paradigm may help vital applications meet real-time temporal constraints by bringing cloud-like services closer to IoT devices. Real-time circumstances like smart healthcare suffer from fog model resource management and load balancing issues that lower service quality and accelerate energy overuse. This paper proposes a fog-based Smart Infection Detection System (SIDS) architecture for quick pandemic control of air-borne infectious diseases such as COVID-19. This design can be used for particular infectious disease epidemics before the pandemic. The network has three planes: IoT, fog, and cloud. An energy-efficient, nature-inspired Geese Optimization Load Balancing Approach (GOLBA), which uses clustering to handle large user requests and select optimal fog gateways for the computation of task requests, is also proposed. Congestion can be avoided, and this strategy can improve system performance. iFogSim simulations compare the proposed approach against an FNPA and round-robin load-balancing methods to show its usefulness. The analyses show that the GOLBA significantly reduces response time by up to 6.6% and energy utilization by up to 8.9% compared to FNPA and response time by up to 13.7% and energy utilization by up to 15.1% compared to RR, respectively.

Summing up the practical implications of the results for intelligent healthcare monitoring, SIDS can increase testing frequency without pushing patients to attend labs during pandemics. The ONFNS algorithm migrates massive tasks and requests data from IoT sensor nodes to fog nodes, which are closest, more active, and more resourceful, to suit delay-sensitive IoT-based smart healthcare applications' QoS needs. When the intelligent healthcare system is widely adopted, the server station balances cluster load using a proposed method with optimal response time and energy use. The recommended method may be tested on more diverse and massive data

sets and confirmed for utilization in further innovative applications like smart cities, buildings, etc. The study can also focus on learning-based adaptive and intelligent approaches to minimize costs and enhance service quality in IoT-based innovative applications.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

Author A.K.C., proposed idea, and G.G., conduct the research, developed methodology & wrote paper. A.K.C. and G.G. perform data analysis & interpretations of results after simulations. G.G. work on writing review & editing and A.K.C., provide supervision. All authors have thoroughly reviewed the manuscript for publication and have approved the final version.

ACKNOWLEDGMENT

I also wish to thank the Meerut Institute of Engineering and Technology for their intellectual support during the study endeavour.

REFERENCES

[1] D. Mala, A. Anand, and N. K. Trivedi, "A system for monitoring of COVID-19 patients at home based on internet of things and fog computing," *ECS Transactions*, vol. 107, no. 1, 11039, 2022.

[2] N. Mostafa, "Cooperative fog communications using a multi-level load balancing," in *Proc. 2019 Fourth International Conference on Fog and Mobile Edge Computing (FMEC)*, 2019, pp. 45–51.

[3] M. Al-Razgan, M. M. Hassan, and T. Alfakih, "A computational offloading method for edge server computing and resource allocation management," *Journal of Mathematics*, pp. 1–11, 2021.

[4] Q. Fan and N. Ansari, "Towards workload balancing in fog computing empowered IoT," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 1, pp. 253–262, 2018.

[5] K. N. Tun and A. M. M. Paing, "Resource aware placement of IoT devices in fog computing," in *Proc. 2020 International Conference on Advanced Information Technologies (ICAIT)*, 2020, pp. 176–181.

[6] J. S. Kshatri, D. Bhattacharya, S. Giri, I. Praharaj, S. K. Palo, S. Kanungo, J. Turuk, J. Ghosal, T. Bhoi, M. Pattnaik, and H. Singh, "Analysis of the COVID-19 testing parameters and progression of the pandemic at the district level: Findings from the ICMR Hundred Million Test (HMT) database during the first wave in India," *International Journal of Infectious Diseases*, vol. 122, pp. 497–505, 2022.

[7] H. A. Khattak, G. Ahmed, H. Arshad, A. M. Sharif, S. Jabbar, and S. Khalid, "Utilization and load balancing in fog servers for health applications," *EURASIP Journal on Wireless Communications and Networking*, vol. 2019, no. 1, pp.1–12, 2019.

[8] A. Yasmeen, N. Javaid, O. U. Rehman, H. Iftikhar, M. F. Malik, and F. J. Muhammad, "Efficient resource provisioning for smart buildings utilizing fog and cloud-based environment," in *Proc. 2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC)*, 2018, pp. 811–816.

[9] E. P. Pereira, E. L. Padoin, R. D. Medina, and J. F. Méhaut, "Increasing the efficiency of fog nodes through of priority-based load balancing," in *Proc. 2020 IEEE Symposium on Computers and Communications (ISCC)*, 2020, pp. 1–6.

[10] R. Beraldi, C. Canali, R. Lancellotti, and G. P. Mattia, "Distributed load balancing for heterogeneous fog computing infrastructures in smart cities," *Pervasive and Mobile Computing*, vol. 67, 101221, 2020.

[11] I. Fatima, N. Javaid, M. N. Iqbal, I. Shafi, A. Anjum, and U. U. Memon, "Integration of cloud and fog based environment for effective resource distribution in smart buildings," in *Proc. 2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC)*, 2018. pp. 60–64.

[12] D. A. Chekired, L. Khoukhi, and H. T. Mouftah, "Queuing model for evs energy management: Load balancing algorithms based on decentralized fog architecture," in *Proc. 2018 IEEE International Conference on Communications (ICC)*, 2018, pp. 1–6.

[13] M. Zahid, N. Javaid, K. Ansar, K. Hassan, M. K. U. Khan, and M. Waqas, "Hill climbing load balancing algorithm on fog computing," in *Proc. the 13th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC-2018)*, 2018, pp. 238–251.

[14] A. A. Butt, S. Khan, T. Ashfaq, S. Javaid, N. A. Sattar, and N. Javaid, "A cloud and fog-based architecture for energy management of smart city by using meta-heuristic techniques," in *Proc. 2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*, 2019, pp. 1588–1593.

[15] Z. A. Khan, A. A. Butt, T. A. Alghamdi, A. Fatima, M. Akbar, M. Ramzan, and N. Javaid, "Energy management in smart sectors using fog-based environment and meta-heuristic algorithms," *IEEE Access*, vol. 7, pp. 157254–157267, 2019.

[16] M. K. Hussein and M. H. Mousa, "Efficient task offloading for IoT-based applications in fog computing using ant colony optimization," *IEEE Access*, vol. 8, pp. 37191–37201, 2019.

[17] V. Divya, "Intelligent deep reinforcement learning based resource allocation in fog network," in *Proc. 2019 26th International Conference on High Performance Computing, Data and Analytics Workshop (HiPCW)*, 2019, pp. 18–22.

[18] I. M. Ali, K. M. Sallam, N. Moustafa, R. Chakraborty, M. Ryan, and K. K. R. Choo, "An automated task scheduling model using non-dominated sorting genetic algorithm II for fog-cloud systems," *IEEE Transactions on Cloud Computing*, vol. 10, no. 4, pp. 2294–2308, 2020.

[19] K. K. R. Hameed, K. Munir, S. ul Islam, and I. Ahmad, "Load-balancing of computing resources in vehicular fog computing," in *Proc. 2020 3rd International Conference on Data Intelligence and Security (ICDIS)*, 2020, pp. 101–108.

[20] J. Yan, J. Wu, Y. Wu, L. Chen, and S. Liu, "Task offloading algorithms for novel load balancing in homogeneous fog network," in *Proc. 2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, 2021, pp. 79–84.

[21] A. Asghar, A. Abbas, H. A. Khattak, and S. U. Khan, "Fog based architecture and load balancing methodology for health monitoring systems," *IEEE Access*, vol. 9, pp. 96189–96200, 2021.

[22] J. Wan, B. Chen, S. Wang, M. Xia, D. Li, and C. Liu, "Fog computing for energy-aware load balancing and scheduling in smart factory," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4548–4556, 2018.

[23] A. Alzeyadi and N. Farzaneh, "A novel energy-aware scheduling and load-balancing technique based on fog computing," in *Proc. 2019 9th International Conference on Computer and Knowledge Engineering (ICCKE)*, 2019, pp. 104–109.

[24] G. Li, J. Yan, L. Chen, J. Wu, Q. Lin, and Y. Zhang, "Energy consumption optimization with a delay threshold in cloud-fog cooperation computing," *IEEE Access*, vol. 7, pp. 159688–159697, 2019.

[25] M. Kaur and R. Aron, "Energy-aware load balancing in fog cloud computing," *Materials Today: Proceedings*, 2021.

[26] M. I. Bala and M. A. Chishti, "Offloading in cloud and fog hybrid infrastructure using iFogSim," in *Proc. 2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, 2020, pp. 421–426.

[27] X. Xu, S. Fu, Q. Cai, W. Tian, W. Liu, W. Dou, X. Sun, and A. X. Liu, "Dynamic resource allocation for load balancing in fog environment," *Wireless Communications and Mobile Computing* 2018.

[28] H. M. K. K. M. B. Herath, G. M. K. B. Karunasena, H. M. W. T. Herath, H. D. N. S. Priyankara, B. G. D. A. Madushanka, and W. R. De Mel, "Integration of IoT and Fog computing for the development of COVID-19 cluster tracking system in urban cities," *Computational Intelligence for COVID-19 and Future Pandemics: Emerging Applications and Strategies*, 2020, pp. 145–169.

[29] G. Goel and A. K. Chaturvedi, "A systematic review of task offloading & load balancing methods in a fog computing environment: major highlights & research areas," in *Proc. 2023 3rd International Conference on Intelligent Communication and Computational Techniques (ICCT)*, 2023, pp. 1–5.

[30] G. Goel and A. K. Chaturvedi, "A comprehensive review of QoS aware load balancing techniques in generic & specific fog deployment scenarios," in *Proc. 2023 International Conference on Computational Intelligence and Sustainable Engineering Solutions (CISES)*, 2023, pp. 983–987.
[31] A. Asghar, A. Abbas, H. A. Khattak, and S. U. Khan, "Fog based architecture and load balancing methodology for health monitoring systems," *IEEE Access*, vol. 9, pp. 96189–96200, 2021.
[32] H. Gupta, A. V. Dastjerdi, S. K. Ghosh, and R. Buyya, "iFogSim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments," *Software: Practice and Experience*, vol. 47, no. 9, pp. 1275–1296, 2017.
[33] A. U. Rehman, Z. Ahmad, A. I. Jehangiri, M. A. Ala'Anzy, M. Othman, A. I. Umar, and J. Ahmad, "Dynamic energy efficient resource allocation strategy for load balancing in fog environment," *IEEE Access*, vol. 8, pp. 199829–199839, 2020.