International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys.13, No.1 (Apr-23)

http://dx.doi.org/10.12785/ijcds/130158

Load Balancing in Fog Computing: A Detailed Survey

Kavitha M S[1], Naidila Sadashiv[2] and Dilip Kumar S M[3]

[1]*Silicon Realization Group, Synopsys, Bangalore, India*
[2]*Department of Computer Science & Engg, JSS Academy of Technical Education, Bangalore, India*
[3]*Department of Computer Science & Engg, University Visveswaraya College of Engineering, Bangalore, India*

Received 6 Mar. 2022, Revised 12 Jan. 2023, Accepted 6 Feb. 2023, Published 16 Apr. 2023

Abstract: Fog Computing (FC) enhances the proficiency and performance of cloud with the objective of bringing selected computing capabilities to the network edge to reduce latency, enhance location awareness, provide mobility support, etc. In the FC environment, resources could not be utilized completely due to the wide variation in execution duration and specifications for computing nodes. Therefore, Load Balancing (LB) for the computing nodes in the FC environment is an essential aspect that avoids the situation of under- or over-loaded fog nodes during the execution of Internet of Things (IoT) applications, especially. This paper starts by presenting several FC architectures, their comparisons and LB strategies in different computing environments. An empirical survey of the existing LB mechanisms has been presented along with trade-offs in their performance metrics. Various possible LB metrics have been investigated. Finally, potential challenges, case studies and future directions on LB in FC environments are presented.

Keywords: Fog Computing, Load Balancing, Architecture, Computing Environments

1. INTRODUCTION

With the proliferation of intelligent devices, it is envisioned that industrialists and researchers can predict risks and take precautions ahead of time to conserve valuable resources. Many new applications have emerged from the Internet of Things, including smart parking, traffic control, smart cities, Internet of Vehicles (IoV), smart meters, smart grids, and greenhouses. Since these real-time applications involve computations, a demand exists for resources, and they generate an enormous amount of data. As a result, networks should be able to cope with a wide range of networking issues related to IoT, including heterogeneity, congestion, routing, energy conservation, reliability, scalability, and Quality of Service (QoS).

With the inefficient usage of cloud resources, need for high bandwidth, mobility issues, and federated infrastructures, some drawbacks persist, including high latency, increased congestion, and sizeable idle energy consumption. Furthermore, even though the IoT paradigm provides perpetual connectivity to objects anywhere and anytime, this connectivity is ineffective if the data gathered and sensed are not used on time. Researchers have attempted to address the earlier needs of emerging compute-intensive applications by exploring a new computing approach called fog computing. Fog computing extends cloud computing by enabling end-user devices to digitally connect with data centers to perform processing, storage, and networking. Fog resides in the middle of the cloud and IoT devices. Protects data, accessibility, scalability, and location awareness and provides high QoS for time-critical applications such as telemedicine with low latency. IoT devices sense data transmitted to fog nodes in a fog environment. Smart devices, smart routers, and gateways are used as data hubs in fog architectures to streamline data processing. The resulting reduction in data sent to the cloud is significant. With an increasing rate of data generation, some fog nodes become overloaded, leading to a significant delay in the delivery of services. Fog nodes' computational load is directly proportional to delivery time. Therefore, a heavy processing load on fog nodes results in a longer delivery time. Coordination among fog nodes is vital to resolve this type of issue, and specific nodes should be able to delegate tasks to lesser-overloaded nodes if they are overloaded. This is required both on physical nodes as well as on virtual machines. This LB mechanism disperses the load evenly across virtual machines and hosts. An optimal load balance among fog nodes will minimize resource consumption and response times. Fog employs two approaches for implementing load balancing, viz static and dynamic.

A. Search Criteria

The abstracts of all searches included the keywords "load balancing" and "fog computing". Nonetheless, it is a common method and takes a lot of time. Many keywords and synonyms are being used to match our results including "load distribution" and "load scheduling". The study includes the works that have been carried out between 2015

E-mail address: kavithas@synopsys.com, naidila@jssateb.ac.in, dilipkumarsm.uvce@gmail.com

to 2022.

B. Publications in Load Balancing in Fog Computing

Figure 1 gives the percentage of papers reviewed from different sources in which majority of works on LB have been appeared in IEEE, Elsevier & Springer journals.

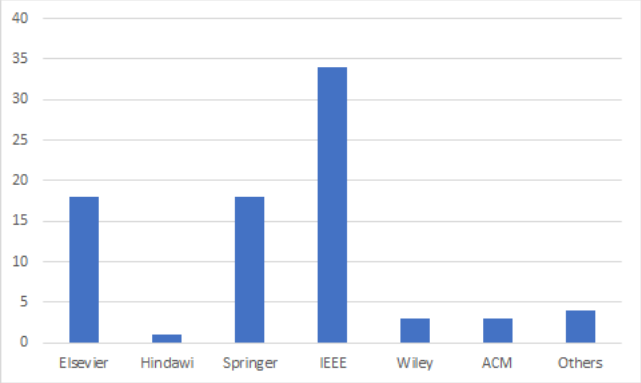

Figure 1. Publications in Load Balancing in Fog Computing.

C. *Contributions of the Paper*

The contributions of this paper are summarized as follows:

1) A comparative study of different fog architectures.
2) An in-depth review of the different load balancing algorithms, strategies, techniques and methods in FC.
3) Provide a detailed view on categorization of works on load balancing in fog environment based on performance metrics.
4) A brief description of challenges in fog based load balancing is presented.
5) Finally, a discussion on case studies of fog computing in real time applications.

Figure 2 depicts the structure of this paper. Following is an outline of the remainder of the paper:
Section 2 presents a brief overview of fog architectures. Section 3 discusses various load balancing strategies employed in computing environments. Subsection provides a brief on different load balancing algorithms Section 4 provides classification of recent works on load balancing in fog infrastructures. Section 5 reviews the various performance metrics for measuring a new load balancing technique or algorithm. Section 6 summarizes the various challenges existing in fog load balancing. Section 7 outlines some of the case studies of fog real time applications. Finally Section 8 concludes this paper.

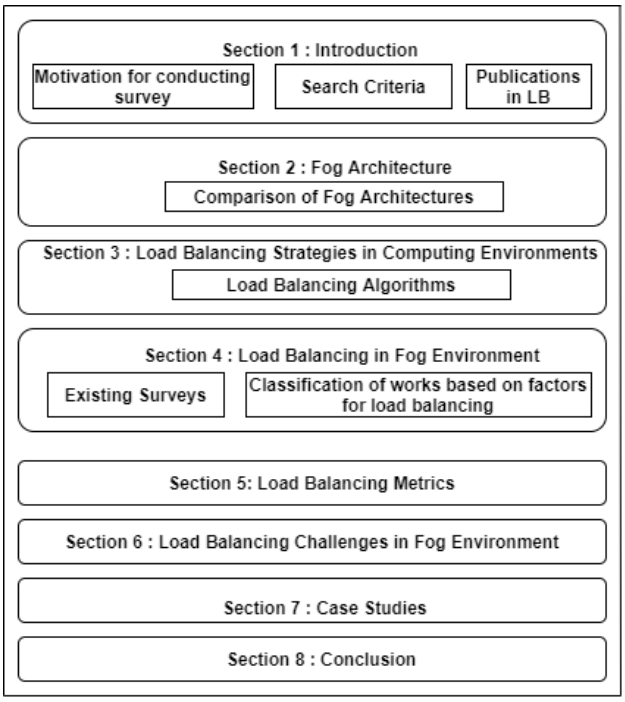

Figure 2. Structure of This Paper.

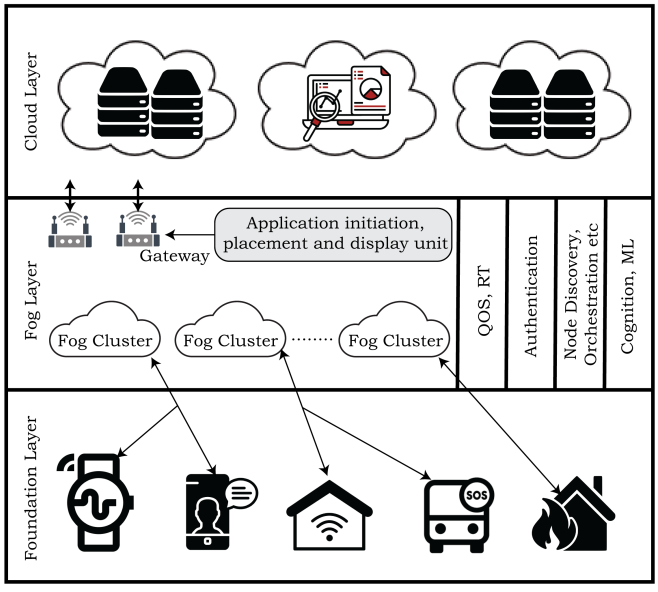

Figure 3. A Typical Fog Architecture.

2. Fog Computing Architecture

The fog computing model extends the traditional cloud model in which applications are developed and implemented across several network layers. According to the literature review findings, the fog architecture is composed of three layers, as depicted in Figure 3.

The *IoT layer* is the foundational layer of the fog architecture. The devices in this layer sense and collect data from heterogeneous geographic locations. Examples include smartphones, sensors, smart cars, readers, smartcards, etc. *Fog nodes* are the devices in the fog layer collectively referred to as the *fog layer*. Access points, routers, gateways, base stations, and fog servers are the devices found in this layer. A fog node resides between cloud data centers and end-user devices on a network; it may be static, e.g., located in a coffee shop, airport, or mobile.

The *cloud layer* at the top contains large storage devices and high-speed servers that can provide high computing performance, computation analysis, and data storage permanently for users' backup and to provide access to personal data. As a rule, information not required in the client vicinity is stored in a cloud layer. OpenFog RA [16] describes multiple perspectives and stakeholder views needed for a specific fog computing deployment or scenario. OpenFog Consortium has defined the fog architecture, which encompasses several perspectives: performance, security, manageability, data analytics, control, business applications, and cross-fog applications. Three views are identified and discussed in detail: system view, software view, and node view.

Table I provides a detailed comparison of some of the fog architectures as proposed by the different authors. From various authors viewpoints, application specific fog architectures are discussed below:

Talaat et al. [75] proposed a 3-layer IoT-Fog architecture for healthcare system. The middle fog layer, which handles the requests forwarded by the IoT layer, is comprised of 2 main modules: Load Balancer Agent (LBA) and Adaptive Weighted Round Robin (AWRR). LBA decides the suitable fog server for servicing the request. AWRR employs a reinforcement learning-based Q-learning model to allocate fog nodes and migrate service requests. Neto et al. [56] proposed architecture with a fog management layer responsible for distributing the load based on the statistics of the node maintained in the tables and also considered the Tenant Maximum Acceptable Delay (TMAD), which is the permissible delay for any tenant. But this approach does not consider the resource requirements of a tenant and only concentrates on the allowable delay component of the tenant. Manju et al. [52] proposed a 4-layer fog architecture consisting of four layers: IoT users, unreliable fog nodes, reliable network resources, and cloud datacenters. Each location's controller node is updated on the idle resources available at that location and nearby sites. The controller decides the suitable fog node to service the request based on these updates. Min-Min algorithm is implemented to achieve the same. Osanaiye et al. [59] have proposed a 3-layer architecture with APIs interfacing the adjacent layers. APIs for the Orchestration layer perform analysis, planning, resource allocation, and enforce decision-making. By hiding the heterogeneity of platforms, the fog abstraction layer exposes a uniform and programmable interface that reveals CPU, memory, and network resources seamlessly through a generic API system.

Verma et al. [79] have proposed a 3-tier architecture with the ground tier consisting of IoT devices, and the fog layer acts as a middleware with each fog node comprising a Fog Server Manager (FSM) that serves the requests by looking at the availability of data and load handling capacity. When the primary fog server cannot fulfill a request, it will pass to the adjacent node, the edge network, and finally to the cloud if the request cannot be fulfilled anywhere. Aazam et al. [2] constructed a layered architecture of fog by defining the functionalities of the fog layer as far as extracting and processing data from IoT devices and finally uploading the preprocessed secured data to cloud storage. Zakria et al. [93] proposed a three-tier architecture with numerous smart buildings with controllers at the bottom layer, a fog network with virtual machines, a microgrid at the middle layer, and a cloud at its top layer. Several smart buildings are connected to fogs, and controllers in each cluster control supply and demand from or to smart buildings.

According to Zhu et al. [94], video applications and services were processed and transmitted using fog computing, including proxy-aided rate adaptation and intelligent caching for on-demand video streaming. Real-time real-world video surveillance for surveillance cameras will thus improve the Quality of Experience (QoE) of the virtual desktop infrastructure. Truong et al. [78] presented a novel Vehicular Adhoc Network Architecture FSDN, which combined two emerging technologies, Software Defined Networking (SDN) and FC, for low latency and location-aware services that could satisfy future VAN requirements.

Gazis et al. [24] introduced a flexible operating platform that caters to the operational requirements of a fog computing infrastructure in an industrial context by providing an end-to-end management capability. Marbukh et al. [53] shows a "bird view" architecture of Fog computing architecture as defined by Marbukh, National Institute of Standards and Technology (NIST). Bonomi et al.[10] have proposed a Fog architecture that represents a simplified version of the idealized infrastructure that can adapt to future IoT applications. The Smart Fog architecture proposed by Kimovski et al. [45] consists of three distinct layers: (1) Cloud layer, (2) Fog layer, and (3) IoT layer. As SmartFog uses Cloud and IoT components, it can independently evolve and allow for high degrees of interaction between both. Chun et al. [15] have proposed the Fog architecture based on a publish/subscribe model. Fog nodes communicate based on a topic-based publish/subscribe model, where the publisher sends a message to a subscriber with a 'topic', and the subscriber receives the message if the issue is of interest.

Guibert et al. [30] proposed a Content-Centric Network (CCN) based Fog model. The CCN elements are spread in a three layer scheme. The preprocessing layer, which preprocesses the received data with some caching policies, is followed by the orchestration layer, a controller for other layers. Finally, data packets are forwarded using CCN

TABLE I. **Comparison of Fog Architectures**

Author	# Tiers	Technique/Algorithm Employed	Description	# Nodes	Advantages
Talaat et al. [75]	3	Reinforcement learning based Q-learning model	Load Balancer Agent (LBA) and Adaptive Weighted Round robin (AWRR) with each fog server's cache size, RAM size, CPU usage and adaptive weight (AW) are considered	2 OL, 2 UL and 3 balanced nodes	Optimal migration from overloaded nodes to underloaded nodes
Neto et al. [56]	3	Multi-tenant load distribution algorithm	Fog management layer (FML) distributes load based on the nodes statistics table by considering Tenant maximum acceptable delay (TMAD)	20 nodes (with 5 tenants)	IoT end-users(Tenants) are serviced as per their plan
Manju et al. [52]	4	Min-Min algorithm employed	Fog layer is comprised of unreliable fog nodes and reliable network devices such as routers, gateways etc which act as controllers to forward the user requests to suitable fog nodes	5 datacenters with 5 fog nodes each	Perfroms better than RR and other priority algorithms, Suitable for a small number of cluster nodes
Osanaiye et al. [59]	3	Abstraction and Orchestration layer APIs providing a seamless interface for resource management	Providing a layer of fog orchestration with small software agents called foglets that allow fog nodes to monitor their current state	NA	Probing, analyzing retrieved data and efficient resource management with the help of APIs
Verma et al. [79]	3	Real-time efficient scheduling (RTES)	Fog Server master and fog co-processors in each region ensure availability of fog nodes for user requests	100 tasks per vm	Maximum throughput with minimum execution time
Aazam et al. [2]	6	Specific layers with individual functionality	Functionality is split across different layers for monitoring, preprocessing, temporary storage, security and transport of data	NA	Enhanced modularity and hence easy to implement
Zakria et al. [93]	3	Round Robin (RR), Throttled and Shortest Remaining Time First (SRTF) algorithms	Fog nodes are connected to cluster of small buildings, Controller in each cluster manages demands of each cluster	6 (3 regions with 2 fog nodes and 2 clusters in each region)	Increased resource utilization and reduced cost

standard exchanges with cloud and fog at the forwarding layer.

3. Classification of Load Balancing Strategies in Computing Environments

Load balancing is perhaps the central aspect of any computing environment. Before a problem can be executed on a system, the work must be partitioned among different processors. Due to uneven processor utilization, load imbalance can happen, leading to degraded performance. Hence, it is required to have appropriate load balancers in the environment. A load balancer prevents servers from becoming overburdened by requests. It can be either a software device or a hardware device. It distributes the network traffic between its servers using an algorithm. Load balancing is two-fold. A static load balancing algorithm does not adjust traffic routing. It distributes traffic evenly among all the servers in a group, either in a specified order or at random. Dynamic load balancing makes use of algorithms that take into account the state of each server and distribute traffic accordingly. Algorithms that simulate dynamic workload redistribution continuously monitor changes in load and alter the load accordingly. They can be classified as *adaptive* or *non-adaptive* and *centralized* or *distributed*. Most of the literature work classifies load balancing strategies based on the type of algorithms used, the different load balancers used, based on system state, and based on initiation process [26, 80]. A pictorial representation of the different load balancing strategies is depicted in Figure 4.

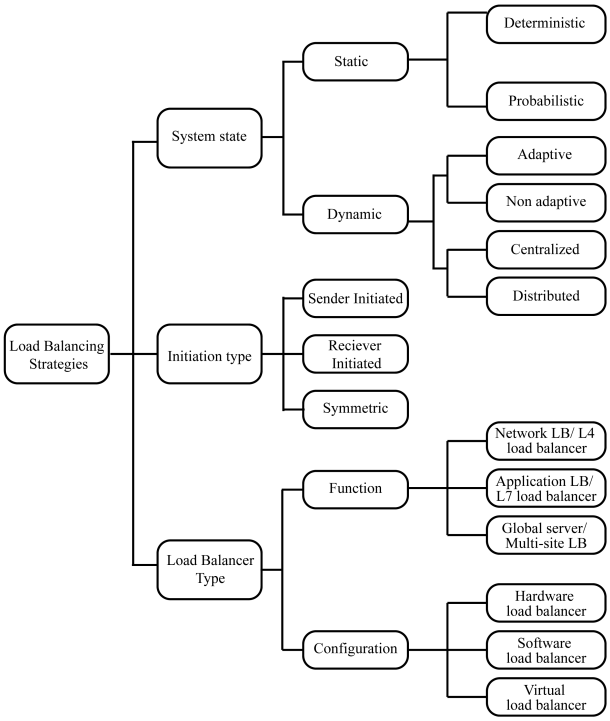

Figure 4. Classification of Load Balancing Strategies.

1) **Based on System State:** Some of the algorithms consider the current state of the system. Parameters such as the system's memory, the storage capacity of the nodes, and processing power can be used to determine the state of a system. Depending on the system's state, algorithms can be further classified as *static* and *dynamic*.
   a) **Static** - Static load balancing algorithms rely on the assumption that all the factors influencing their decisions, such as job characteristics, communication networks, and node locations, are known in advance. Static algorithms are classified as deterministic and probabilistic algorithms. A deterministic or probabilistic load balance is determined during compilation phase, and it remains constant throughout runtime phase. In addition to being easier to implement, this approach involves minimal runtime overhead.
      i) *Deterministic* - These algorithms take into account the properties of nodes and characteristics of the processes to be scheduled.
      ii) *Probabilistic* - The probability algorithms formulate simple rules for the placement of processes by considering static attributes of the system (e.g. nodes, processing capacity, topology etc).
   b) **Dynamic** - Dynamic load-balancing algorithms attempt to make smarter decisions based on runtime state information. A centralized location may make global decisions, or multiple distributed locations can share the responsibility. It results in a better performance compared to static load balancing. They can be classified as Adaptive and Non-adaptive.
      i) *Adaptive* - These algorithms adapt their parameters or even their scheduling policy itself to take account of the global state of the system. Previous decisions and changes in the environment will affect scheduling decisions, which will consider past and current performance.
      ii) *Non-adaptive* - Unlike adaptive balancing, in non-adaptive balancing the parameters remain untouched by the past behavior of the system.
      iii) *Centralized* - A central global controller must have knowledge of the whole fog infrastructure and IoT requirements. Load balancing in such architecture with a single control point can be implemented easily. However, it lacks scalability, fault tolerance and can become a performance bottleneck if not maintained [25, 14].
      iv) *Distributed* - All nodes participate in load-balancing decisions in this approach. The cost of obtaining and maintaining state-of-the-art information for the system becomes prohibitive for each node. The state information is shared among the nodes that manage their resources and allocate tasks in their queues to other nodes. Frequent information exchange between processors results in communication overhead [25, 48].
2) **Based on Initiation Process:** As stated by Rathore et al. in [64], depending on the current load on the node and the originator of the request, load balancing techniques can be classified as:
   a) **Sender Initiated** - A specific task periodically checks for load on the nodes and if an overloaded node is found, it distributes the load evenly to less loaded nodes.
   b) **Receiver Initiated** - An idle processor pulls the load from a busy processor. But both sender initiated and receiver initiated are not mutually exclusive in the computing environment.
   c) **Symmetric** - This approach employs a combination of both push and pull. The Linux scheduler and the ULE scheduler available for Free BSD systems employ this approach.

   Sender and receiver initiated processes employ different policies for selection of nodes for load transfer. A dynamic algorithm considers the current state

of the system, as already mentioned. Some of the policies are:

a) **Information policy** - This policy keeps track of all resource information in the system. Information about the nodes is collected through agents, centralized polling, and broadcasting. The time during which data should be collected also plays a significant role in this policy.
b) **Location policy** - This policy finds an underloaded node and moves the tasks to them for processing. Several approaches can be used to determine the destination node for task migration, including probing, negotiation, and randomization.
c) **Transfer policy** - This policy identifies a task to be transferred to other nodes in 2 approaches. Either it moves all current tasks or the last task received at the node.
d) **Selection policy** - With this policy, tasks transferring from one node to another are selected from those that will cause the most negligible overhead, the fewest number of nonlocal system calls, and the shortest implementation time.

3) **Based on Types of Load Balancer:** For specific network issues, several load balancing techniques are available, including SQL Server load balancing for relational databases, DNS server load balancing to ensure domain name functionality across multiple geo-locations, and global server load balancing for troubleshooting. They can be categorized based on their *functionality* and *configuration* as follows:

a) **Based on Function:**
i) **Network Load Balancer/Layer 4 (L4) load balancer:** The L4 load balancer does not consider application-level parameters such as the type of content, cookies, headers, locations, application behavior, etc. Instead of inspecting the contents of discrete packets, it translates network addressing without inspecting packets. This allows it to direct traffic based solely on network layer information.
ii) **Application Load Balancer / Layer 7 (L7) Load Balancer:** A L7 load balancer distributes requests based on different parameters. The L7 load balancer evaluates variables such as HTTP headers and SSL sessions, as well as a wide range of data, and distributes loads according to several factors.
iii) **Global Server Load Balancer (GSLB)/Multi-site Load Balancer:** GSLB ensures an efficient global load distribution without compromising end-user experiences, by extending the L4 and L7 capabilities across various data centers.

b) **Based on Configurations:**
i) **Hardware LB**: Essentially, this is a piece of hardware on-premises that distributes traffic on several servers. While they can handle a lot of traffic, these systems are quite expensive and moderately flexible.
ii) **Software LB** : These are computer applications that are installed on a system and function similar to hardware load balancers. They map virtual IP addresses (VIPs) to Direct Internet Protocol addresses (DIPs), which are part of a cloud service pool of workload balanced VMs within the datacenter.
iii) **Virtual Load Balancers**: A hardware load balancer and an application running on a virtual machine make up the virtual load balancer. Virtualization mimics the software-driven infrastructure, as software applications are executed on virtual machines to redirect traffic appropriately. Although not equipped with a central management system, these load balancers present similar challenges to in-house physical balancers, including less automation and fewer scalability options.

A. *Load Balancing Algorithms*

Algorithms are designed as static and dynamic algorithms based on the system state, as mentioned earlier. One of the major aspects of the design of load balancing algorithms is determining the appropriate load index to use. The load index can be used to determine the node's status in terms of their availability for accepting additional workload at every instant. A multi-core system load index depends on the number of cores per node and the computing power of each node. Determining a load index does not rely on the same set of parameters.

It varies as per the algorithm designer's needs. Different parameters are considered to calculate the load index, including CPU occupancy, memory usage, system I/O use, and network bandwidth usage. Algorithms are designed by considering these. Early researchers conducted experiments using some of the classical load balancing techniques like Round Robin, Least Connection, Throttled, etc. Table II presents a brief view of classical algorithms.

Random allocation, Round-robin, etc., are classic algorithms that solely use different network parameters to determine where incoming data should be forwarded, without consulting any other components of the computer system such as the quantity of data being collected by the applications and databases.

TABLE II. Classical Load Balancing Techniques.

Author	Algorithm	Type	Description	Advantages	Disadvantages
[70], [19]	RR algorithm	Static	Using the available servers' list, it forwards each task equally to each of them in the order of arrival	Easy to install, Not much complexity on the programmer's end	Lacks an understanding of resource capacities, priorities and length of tasks, which leads to resource starvation
[31] 2019	Weighted Round robin	Static	Its cyclical assignment method is similar to RR, but in that the node with the highest capacity receives the most requests, despite being similar to RR in other aspects	More requests can be sent to servers with higher capacity and handling load	Capacity calculations can be difficult in some situations, for example if the packet size varies
[73] 2018	Least connection	Dynamic	Servers with the fewest active transactions are selected	It prevents server overload by monitoring the number of connections	Counts current connections without taking the server capacity into account
[73] 2018	Weighted Least connection	Dynamic	Keeps a weighted list of active connections between application servers	To prevent overloading and crashing, both the server's capacity as well as current connections are taken into account	Suitable for clusters only
[35] 2012	Weighted Active Monitoring	Dynamic	Based on the processing power of each server, it assigns weight to each in real-time	Optimized response time and processing time of data	The selection of a suitable server does not take into account the server's capacity
[60]	Min-Min algorithm	Static	It considers all unmapped tasks and finds minimum completion time for each task and maps it to the corresponding server	Minimizes makespan	Larger tasks starve for longer period
[60]	Max-Min algorithm	Static	Finds set of minimum completion times M from unmapped tasks and assigns maximum completion time from M to resembling machine	Gives a much better performance than Min-Min, Makespan reduced by executing tasks parallely	Waiting time of smaller tasks is increased and leads to starvation
[60]	LBMM algorithm	Dynamic	LBMM utilizes Min-Min for the first phase, then reschedules tasks to use the underutilized resources for the second phase	Minimizes makespan and improves response time, Utilizes resources effectively	Average response time for smaller tasks is increased
[77] 2018	Tabu search Meta-heuristic method	Dynamic	An integer optimization algorithm to find the size of the tasks in order to make an appropriate assignment for each one	Reduced memory usage and computational costs	Lot os tuneable parameters in the approach consumes more iterations and hence increases time complexity
[82] 2018	Throttled algorithm	Dynamic	In the data center, load balancers manage a table of VM indexes and their locations. In response to the customer's request, the data center locates a suitable virtual machine	Improved response time compared to other algorithms	There is no simulation for a specific workload situation and no limit on the duration

TABLE III. Conglomerate Load Balancing Techniques.

Author	Algorithm	Description	Advantages	Disadvantages
[63]	Central Load Balancing Decision Module technique (CLBDM)	Acts as monitor and impacts on forwarding decisions on load balancers	Dynamic allocation of resources between the VM's, Signals the Load balancer if a server is in trouble and unable to accept new sessions	Having a single point of failure will cause erratic decisions and requests to bounce between nodes, leading to poor performance and an unpleasant user experience
[62]	Randomized algorithm (Balls into Bins via Local search)	Bins(servers) represent nodes in graph and edge weight represent distance between bins. Minimum spanning tree is generated to find suitable server for execution	Workload difference between the nodes is minimized	Best results are seen only when no of tasks is atleast 2000
[21]	Divide-and-Conquer combined with throttled algorithm	Requests from clients are distributed to load balancer and suitable real time requests are forwarded to Request handler (RH) and virtual machine deals with simulation environment	Resource utilization maximized, Total execution time reduced	Not tried to simulate differing workload situations, Task deadlines are not considered
[69], [92]	Stochastic Hill Climbing (SHC) associated with Join Idle Queue (JIQ) algorithm	SHC with JIQ find a suitable virtual machine for the execution of job	Improved response time, Better resource utilization	Not suitable for large no of requests

The work in [63] proposed a centralized load balancing software model that periodically interacts with load balancers and application modules. But this approach of having a centralized decision-making system poses the challenge of a single source of failure, bringing the entire system to a halt. Few works have improvised a base algorithm or combined different algorithms to arrive at their results. For example Stochastic Hill Climbing (SHC) algorithm with Join Idle Queue (JIQ) to find a suitable virtual machine (VM) for igration of load [69]. Table III presents the works considering such conglomerate algorithms.

Apart from the above generic method, some of the works have considered categorizing the algorithms based on the nature of tasks as *Natural Phenomena based* and some based on the concept of intelligent agents and are classified as *Agent based* algorithms. Table IV presents several LB techniques inspired by natural phenomena. Recent works discuss the usage of hybrid algorithms like Cuckoo search with Levy walk distribution, combination of Honey bee with enhanced weighted RR algorithm, and Chaotic social spider algorithm. The chaotic social spider algorithm achieves a minimum makespan and utilizes resources efficiently. The work lacks in identifying trusted nodes for load distribution. Addressing the problem of migration of tasks between VMs leaves the Cuckoo search algorithm a good choice for balancing the loads on nodes.

Some real-time applications like the Smart industry consider the installation of intelligent agents for their smart decision-making systems to automate complex tasks. For example, smart industry applications consider smart shop-floor objects such as machines, conveyors, and products to be associated with terminals. Smart objects are classified into various types of agents who can make autonomous decisions and do distributed cooperation amongst them for discovering the resources and negotiating and balancing the resources. The agents will perform these functions automatically and continuously to fulfill the design objective. The work on agent-based information retrieval is insignificant and suits specific applications like smart factories. Table V gives a detailed view of the load balancing techniques that employ agent nodes for deriving the node statistics for load distribution.

4. Load Balancing in Fog Environment

This section examines the limited number of Fog computing based LB survey works and also classifies existing LB approaches based on different performance parameters.

TABLE IV. Load Balancing Techniques Inspired by Natural Phenomema.

Author	Technique	Objectives	Advantages	Disadvantages
[36] 2019	Cuckoo search along with Levy walk distribution and Flower pollination LB algorithm	Reduce delay and latency issues and increase performance of fog	Have considered 6 different smart communities aka smart home, smart city, smart grid etc, Increased response time, processing time and reduced cost	problem of migration of tasks between VMs and bin packing problem are unresolved
[61] 2019	Modified Honey bee behaviour algorithm with enhanced Weighted RR algorithm	To achieve minimal completion time and to improve CPU processing time	Modified honey bee algorithm for handling prioritized tasks and weighted round robin for non prioritized tasks, Improved system performance and resource utilization, Minimum completion time	Migration costs increase considerably due to shifting of priority tasks to suitable VM, Waiting time increases due to migration
[86] 2018	Chaotic social spider algorithm (CSSA)	To achieve minimum makespan and balanced resource utilization	Better makespan optimization, minimal degree of imbalance compared to algorithmsGa, ACO, PSO and Hybrid fuzzy K-Means++ with colonal selection	No means to identify trusted nodes for migration
[89] 2015	Cuckoo Optimization algorithm	Maximizing resource utilization,Reducing energy consumption	Uses Minimum migration time (MMT) policy for identifying over/under-utilized hosts for VM migration	Security during VM migration is not ensured, Increased response time, SLA violation
[17] 2015	Ant Colony optimization (ACO) integrated with Genetic algorithm (GA)	QoS-GAAC task scheduling algorithm with QoS ensured	User QoS is improved, Resource load balancing is achieved which improves overall performance of the system	Dynamic variation of QoS paramters need to be considered
[85] 2014	Job spanning time and load balancing genetic algorithm	JLGA proposed a new task scheduling method that is intended to achieve load balancing with least makespan	Adoption of greedy algorithm to identify load intensitivity among nodes and hence JLGA has better performance in terms of makespan	Job priority not considered
[51] 2015	Osmosis LB algorithm	Load balancing in both heterongenous and homogenous environments using Distributed hash table (DHT) with chord overlay mechanism	VMs in both homogenous and heterogenous environments are considered, Supportive in reallocating the tasks between adjoining VMs	Very few data centers can support heterogenous lattice. Homogenous lattice expects only non preemptive and indivisible tasks
[18] 2013	Genetic algorithm	Minimizing completion time of given tasks thereby trying to balance load on ndoes	Improved resource utilization, Reduced job time span	Low throughput, Jobs are not prioritized

TABLE V. Load Balancing Techniques Employing Agent Nodes.

Author	Technique	Objectives	Advantages	Disadvantages
[83]	Energy-aware Load Balancing and Scheduling method	An agent-based system to schedule manufacturing clusters in a distributed manner	The problem of scheduling in large-scale job clusters is solved by multi-level and multi-diversified agents based on autonomous decisions	Due to dynamic randomness in scheduling, broadcasting of information is inefficient
[81] 2016	Autonomous Agent Based Shortest path using Dijikstra's algorithm	Minimizing request time and simulation time by finding a shortest path to specific VM	Reduced request time and simulation time	Throughput not guaranteed due to increased search time
[23] 2019	Self-Governing Agent Based Load Balancing Algorithm (SGA-LB)	To increase performance and efficiency	Service time can be reduced by tracking virtual information using internal and external agents	Maintainence of in-house agent and communication to external agent are an overhead
[32] 2015	MapLoad enhanced with live	Load balancing in a distributed and scalable fashion	Agents are able to autonomously and dynamically balance load with partial information about the data center, Considers heterogenous environment	Decision making process of agents is incomplete as they are unable to predict resource usage, VM migration overhead is not taken into account
[29]	An Agent Based Dynamic Load Balancing (ABDLB) approach in which mobile agents play a key role	Aims to reduce server communication costs and to increase load balancing rate	Communication cost from server to server is 10 times better than Centralized server based LB technique, Improved throughput and response time of the cloud	Increase in number of servers induces communication overhead
[84]	OLB + LBMM Scheduling algorithm	Node information is collected by an agent and threshold values of each node are evaluated so that a service node may be assigned	Makes better utilization of resources, Ensures minimum execution time for each task and on a whole, minimum completion time	It is difficult to select the right service node for each incoming task based on some threshold

A. *Existing Surveys on Load Balancing in Fog Environment*

A recent survey by Kaur et al. [42] provides a taxonomy of LB techniques and performance metrics that may impact load balancing in FC environment. The work also highlights that all the existing works have been carried out and tested only in a simulation environment using simulations tools like ifogsim, fognetsim++ etc.; hence, implementing it in a real test bed environment is essential. In [41] Kashani et al. systematically analyze load balancing mechanisms within fog computing in four classifications, which include approximate, exact, fundamental, and hybrid methods. A 3-layer fog architecture, various load balancing metrics, their merits/demerits, evaluation tools/techniques, and finally, the open challenges and future trends of those mechanisms are described. Chandak et al. [14] in their work on the review of load balancing in fog environment have tried to cover some aspects like fog architecture, various load balancing techniques and simulation tools used. The heterogeneity of nodes, their allocation have also been discussed. Ghobaei et al., in their study on resource management approach in fog computing [25] briefly describe the load balancing issue of fog computing. The work provided a taxonomy of the state-of-art resource management mechanisms and classified it into six main fields: application placement, resource scheduling, task offloading, resource allocation, resource provisioning, and load balancing. Traditional approaches to load balancing suffer from high computational overhead, energy consumption, and deadline constraints. The comparison of surveys conducted by different authors has been detailed in Table VI.

TABLE VI. **Comparison of Surveys Conducted by Various Researchers**

Ref Papers	LB Arch	LB Metrics	Class of LB algo	Issues	Tools used	Class of LB on Metrics
[25]	No	Yes	No	Yes	No	No
[14]	Yes	No	Yes	Yes	No	No
[42]	Yes	No	Yes	Yes	Yes	No
[41]	Yes	Yes	Yes	Yes	Yes	No
Our Work	Yes	Yes	Yes	Yes	Yes	Yes

B. *Classification of works based on Performance metrics for Load Balancing in Fog Environment*

This section explores the load balancing issue in fog computing carried out by various researchers. Classification is made based on the different performance metrics. The parameters considered for balancing the load, their experimental results, and appropriate tools and technologies have been summarized.

1) **Load Balancing works considering Bandwidth factor**

Recent research by Baccarelli et al. [6] proposed the convergence of Deep Learning and Fog Computing paradigms that allow IoT devices to generate large volumes of data in real-time in an energy-efficient manner, despite their real-time and resource-limited nature. An Artificial Intelligence (AI) platform named Learning-in-the-Fog (LIFO) has been developed, which can detect changes in the environment and automatically adjust the available computing resources and networks in response. The work in [54] has proposed a novel Mixed Integer Linear Programming (MILP) optimization model encompassing an objective function to limitate bandwidth cost and provide LB. Investigation of different scenarios to evaluate the experiment is ensured by leveraging SDN. The study considers both parameters: servers' CPU processing capacity level and links' bandwidth. Each parameter is prioritized by assigning a weight factor and hence manages the load by minimizing the queuing delay of links. The work considers both homogeneous and heterogenous resource demands generated by the cluster points. Bandwidth, no of servers and links constitute the different resources. Variation in the bandwidth range and connections happen dynamically for heterogeneous resource demands, thereby adjusting the weight factor for the corresponding component.

2) **Load Balancing works considering Energy Consumption factor**

The Fog Computing Architecture of Load Balancing (FOCALB) has been proposed by Kaur et al. [44] in the context of scientific workflow applications. In addition, they offered a hybrid load balancing algorithm combining Ant Colony Optimization (ACO), Grey Wolf Optimization (GWO) and Tabu search. Using the proposed model, load balancing is implemented at the fog layer. Load scheduling is performed at the fog nodes when tasks are created, and load balancing is performed at the local controller in fog clusters. In addition to reducing the execution time and implementation cost, FOCALB minimizes the energy consumption at fog nodes. Talaat et al. [50] proposed an energy efficient fog computing for healthcare industry. The proposed strategy is designed to improve the energy efficiency of fog devices by employing Improved Round Robin (IRR) and dynamic voltage and frequency scaling algorithms. A novel energy-aware load balancing algorithm was proposed by Kaur et al.'s. [43] work to enhance service quality and reduce latency. Based on a comparison with the Tabu search method, the proposed algorithm aims to improve execution time and reduce energy and cost consumption in fog environments. By distributing processes and data among fog nodes and servers, Oma et al. [58] propose a tree-based fog computing model to reduce the energy consumption by IoT nodes. Anees et al. [65] proposed a dynamic energy-efficient resource allocation strategy which is composed of various modules to collect the end user requests and statistics of resource availability, a resource scheduler for scheduling the processes and accountability of power consumption for each of the processes through resource on/off mechanism. Compared to existing DRAM schemes, the proposed scheme reduces the energy consumption and computation cost by 8.67 percent and 16.77 percent, respectively. The work in [47] discusses that context-awareness is improved with human-driven data analytics and resource-sharing network adaptability. A cluster of fog nodes could enable task offloading, thereby reducing the energy consumption and latency for end-user devices [66]. In the work, [74] a fuzzy load balancer has been devised by Simar et al. by using a variety of designs and tunings of fuzzy logic control algorithms. Network link analysis has been carried out to plan traffic flow with a fuzzy logic algorithm. The research shows that by reducing the number of intervals, reducing overhead in provisioning, and enhancing responsiveness, a 3-level fuzzy design for load balancing in fog zones is energy efficient.

3) **Load Balancing works considering Availability and Latency factors**

In [76], the Efficient Load Balancing Strategy (ELBS) scheme has been proposed with various real-time scheduling algorithms like Fuzzy and probabilistic neural networks (PNN). The entire work is carried out by splitting the task into multiple modules, and appropriate algorithms are employed in each module to achieve the result. This method achieves low latency while requiring a relatively low number of migrations. Using reinforcement learning and genetic algorithms, a Load Balancing and Optimization Strategy (LBOS) using a dynamic resource allocation method has been proposed by Talaat et al. in [75]. The work mainly fo-

cuses on healthcare systems that require a constant response with no delay in their service due to the exigency of the requests, privacy of an individual's data, and accuracy. A 3-layer fog architecture has been proposed where a fog layer is comprised of a Load balancer agent (LBA) and Resource Allocator (RA). A fog server's (FS) availability is checked based on the Adaptive weight (AW), which is calculated by considering cache size, RAM size and CPU usage. A distributed peer-to-peer (P2P) load balancing algorithm, LL(F, T), was proposed by Bellaldi et al. [8]. If a fog node's current load exceeds T, a random selection is made overall F fog nodes. In preliminary simulations and mathematics, it has been demonstrated that tuning T very close to the node saturation condition maximizes performance on par with its classical implementation using a single global scheduler. This eliminates the need for a time-consuming probing phase before each job execution, a critical component of fog applications that require low delays. Singh et al. [72] has put forth three algorithms: minimum distance (MD), minimum load (ML), and minimum hop distance and load (MHDL) to achieve low communication latency between the fog-based micro data centers (mdcs) in the fog environment. Additionally, they presented a load balancing aware scheduling algorithm based on ILP (LASILP). Together, these schemes have achieved optimum response time and network consumption performance.

4) **Load Balancing works considering Mobility and Interoperability factors**

Static resource allocation and dynamic service migration are required to achieve load balancing in a dynamic network. Static resource allocation is supported with the help of virtualization technology which improves resource utilization in a fog environment. Many practical applications are made possible by virtual machines(VMs) built on existing virtualization technologies. Even if VMs require only a few seconds to start up, any delay is unacceptable for tasks that need to be completed immediately. A suitable option to replace VM's in fog nodes is containers. As containers can modify resources while running, this approach allows for dynamic resource adjustment based on the concurrent tasks, particularly useful in fog environments [91, 38]. Xu et al. follow a four-step process to balance the load in a dynamic network [88]. According to the resource requirements of the node type, fog services are classified into several sets, each subdivided into numerous subsets based on the request's start time. The next step is to detect the spare space of all computing nodes to determine whether the computing node is portable to host the fog service, then employed resource units are analyzed through occupation records and then the spare space of all computing nodes. Computing nodes are identified for fog services in the same service subset. Generally, computing nodes with least and most adequate spare space are selected to allocate fog resource units. In addition, workloads from computing nodes with higher resource demands are migrated to computing nodes with lower resource demands.

It represents a significant challenge to allocate and manage resources for IoT devices in an IoV scenario due to the high degree of mobility of the devices [28]. Based on a prediction of mobility patterns in Smart Cities, the work proposed a VM migration procedure based on their highly predictable mobility. The algorithm defines a set of candidate cloudlets based on the user's future position to determine the nearest node for forwarding the request. Further, an Integer Linear Programming (ILP) model is proposed to enhance the placement of VMs within the candidate cloudlet set. Sharma et al. [71] proposed a novel 4-tier architecture with IoT devices at the bottom tier (Tier-1). Workloads are prioritized in the second tier using a Dual Fuzzy Logic algorithm. The third tier (fog tier), a fog node is assigned to a high-priority task. Clustering fog nodes using K-means++ routes requests to the most appropriate clusters. Unhandled requests will be forwarded to the cloud tier. In [88] a system framework for fog computing has been proposed by Xu et al. to implement the dynamic resource allocation method (DRAM). Initially, an analysis of load-balancing is presented for different types of computing nodes. Following that, a load-balancing method is developed that leverages static resource allocation in FC and dynamic service migration. Self similarity-based load balancing (SSLB) method has been proposed by Li et al. in [48] where each fog node is composed of three components: LBMonitor, LBScheduler, and LBMessenger. To balance the workload periodically, they must detect local loads, communicate with other nodes for global information and then make decisions based on this information. Ningning et al. follow a graph repartitioning model to balance the load in a dynamic network [57]. A cluster of VMs and physical nodes are formed. According to the resource distance, task load balancing, and other conditions, the VM nodes provide services to the users by Graph partitioning and clustering. When the system changes graph is repartitioned based on the historical segmentation information. The changes in the system state include the node join and exit, the increase or decrease of resources, and link bandwidth.

5) **Load Balancing works on Resource Management factor**

Beraldi et al. [9] have devised an algorithm for a realistic smart city scenario with dynamic infrastructure. Two distributed load balancing algorithms have been proposed to address this dynamic requirement, which varies over time and space, namely, sequential forwarding and adaptive forwarding. The proposed algorithm is compared with a baseline algorithm without load balancing, in which no load balancing occurs among the fog nodes. A study by Javaid et al. [37] was focused on minimizing latency and enhancing reliability by managing resources in the fog. Resource optimization was explored by considering a fog and cloud environment together. RR algorithm, Equally Distributed Execution algorithm, and the Shortest Job First proposed algorithm are used to estimate request frequency, processing time, and response time. A Fog-2-Cloud framework has been implemented for managing customers' demands with six fog nodes and twelve MicroGrids in residential

buildings. Neto et al. [56] has proposed a multi-tenancy load distribution algorithm (MtLDF) that supports load balancing in fog environments by addressing the delay and priority requirements for multi-tenancy. The algorithm considers the tenants - IoT end-users' maximum acceptable delay and priority requirements. Tenants are sorted based on the plan they have subscribed for and serviced according to their subscription.

6) **Load Balancing works which focus on specific Use cases**

Several other researchers have proposed various schemes and algorithms for managing load in dynamic environments and employed them for certain real time use case scenarios. Ahmed et al. [39] considers the application of fog environment for Internet of vehicles (IoV). The merged system takes advantage of SDN, IoV and fog computing, treating parked vehicles as assistant fog computing nodes. By utilizing SDN controllers and fog managers, the LB in their work proactively balances the load locally and globally. In the work of Xu et al. [87] a heuristic method of scheduling virtual machines has been devised through VM placement by leveraging VM migration in two ways - a resource model and a load balancing model. Baburao et al. [5] studied various service migration strategies, load optimization techniques and load balancing techniques used in fog computing environments. Due to the dynamicity of IoT devices locations, fog nodes are expected to enable service mobility across the servers when IoT devices move between the locations. So, several service migration techniques have been there in existing research like statefull migration for a continous service, workload driven migration which expects a daemon fog node to retain the statistics of resource usage. Zakria et al. [93] in their work, have discussed about the integration of Micro Grid with cloud computing to form a Smart Grid. Fog nodes are selected using a nearest data center service broker policy and the Shortest remaining time first algorithm. SRTF achieves a better cost reduction over RR and Throttled but increased response time and processing time as the number of VMs increase. In literature, many researchers describe the concept of a smart gateway, which can pre-process and trim data before transferring it to the cloud. However, Sarma et al. [68] have concluded that a smart gateway need not do data trimming and preprocessing every time, but instead it can function as a smart load balancer. In addition, it is stated that the performance of Fog and Cloud can be improved by a sophisticated and standard load balancing algorithm. The work of Divya et al. [20] presented a reinforcement learning based load balancer algorithm using a basic tuple $\langle S, A, P, R \rangle$ which learns the load status of the network and distributes it accordingly. A novel dynamic load balancing mechanism for fog computing based on graph partitioning was proposed by Ning et al. [57]. The system state is revisited every time when a new node joins or exits. If the new node's resource is greater than the standard value of virtual machine resources, the node will be atomized into two-virtual machine nodes and a virtual physical node. If the new node's resource is less than the standard value of virtual machine resources, the new node will be cloudization to the smallest connected virtual machine node. The dynamic repartition ends if the request is able to find a suitable VM else it continues. Generally, fog nodes have limited processing capabilities. Therefore, it is likely that they will be overloaded quickly. This situation requires the transfer of certain jobs to other fog nodes of relevance, so the network will be balanced and performs more efficiently. The comparitive study of works discussed above have been detailed in Table VII.

Comparative study of works based on QoS-parameters has been detailed in Table VIII. Certain trade offs exists in some of the works while they achieve their predefined objectives to meet performance metrices. A brief view of the trade offs has been outlined in Table IX.

5. Load Balancing Metrics

A new load balancing strategy or technique developed could be evaluated with various performance parameters. Some of those parameters used to measure an algorithm's efficiency are:

1) *Performance*: A system's effectiveness depends on some of the key load balancer performance metrics like request counts, active connection or flow counts, error rates, latency, number of healthy/unhealthy hosts and rejected or failed connection counts.
2) *Response Time*: Basically, it refers to the time that passes between the time the request is submitted and the time when the first response is delivered. Time it takes to start responding for a user request is an important factor for real time interactive applications [72].
3) *Throughput*: Describes how many tasks or processes are completed on a system within a given period of time. The greater the throughput, the better the response time and efficiency.
4) *Service Uptime*: Placing a critical subservice in the Fog rather than in the Cloud can improve service availability in the presence of hostile environments (e.g., those with intermittent network connectivity). Therefore, service uptime may be another performance metric.
5) *Scalability*: It is the ability to achieve uniform load balancing between increased workloads and decreased workloads by scaling up and down.
6) *Fault Tolerance*: Resource failures (processors/links) are likely to occur frequently in fog environments due to resource limitations, which will adversely affect time-critical applications. Therefore, it is increasingly important to develop techniques to cope with faults. Scheduling replicas of the same job on different nodes can ensure fail-safe behaviour.
7) *Migration Time*: This is a measure of how long it takes a request to travel from an overloaded to an underloaded machine. The shorter the migration time, the better the system's performance [76].

TABLE VII. Comparative study of works on Load Balancing in Fog Environment

Author	Technique	Main Objective	Description	Tools used	Pros	Cons
[65]	DEER Strategy	To Minimize energy consumption and computational cost	Application modelling to evaluate DEER algorithm for performance evaluation parameters	Cloudsim	Reduced energy consumption and computational cost	Not Fault tolerant and varying # of tasks not considered
[22]	LAB Scheme	To reduce computational and communication latency	Base station & IoT device side algorithms for LB on device side & base station side.	Matlab	Overall latency is minimized	Communication latency slightly degrades while balancing the other
[7]	MILP model	The goal is to design a queuing system that models the performance of IoT edge networks based on multiple vacations	Load-Balancing Algorithm for Multiple Gateways in Fog-based IoT	Maple 16 and Matlab	Caching resulted in fast response and reduced energy consumption	There is no consideration of different types of task requests
[83]	Multi-agent model	To reduce complex energy consumption and scheduling problem in large scale job clusters like smart factory	Minimized energy consumption and equipment workload by prioritizing based on their energy intensity	Prototype model	Optimized scheduling and reduced energy consumption for large clusters	Reduced efficiency due to high volume broadcasting data
[54]	MILP based optimization model	To minimize bandwidth cost of routing, maximum link utilization of network links and server utilization	Weights are associated with the components of objective function to prioritize them based on the network situation	AMPL / CPLEX tool	The proposed model can serve as a benchmark to compare the efficiency of fast solution heuristics	Requests are not prioritized.
[76]	ELBS strategy	To reduce average turnaround time and failure Rate	Fuzzy inference system prioritizes the incoming requests and PNN algorithm assigns server for processing the request	iFogsim	Low network Latency and moderate number of migrations	Not fault-tolerant and health care specific
[75]	LBOS using DRAM based on RL & GA	To improve QoS in terms of allocation cost, to reduce response time and to achieve low latency	Load Balancer Agent and Resource Allocator modules allocate suitable fog nodes	Matlab	Efficient resource utilization, continuous availability, reduced power consumption and migrations	Specifically implemented only for health care applications
[4]	Fog nodes with LBSSA scheduler	Prioritizing the requests as real-time, important and time-tolerant to provide reliable services	Adaptive weighted round robin algorithm assign resources to prioritized nodes	Cloudsim	Highly reliable service for time-tolerant requests & efficient resource usage	Important unscheduled real-time requests preempt other tasks
[48]	K-means & task distribution algorithms	To achieve scalability, flexibility, and low latency in large environments	Periodic workload balancing by fog node components	iFogsim	Calculation of load thresholds on nodes and resulting in low latency	Fog topologies with dense computing infrastructures are unsuitable

TABLE VIII. Load Balancing with various QoS Parameters by Researchers

Ref. Papers	Energy Consumption	Latency	Makespan	Exec Time	Response Time	Cost
[47]	Yes	Yes	No	No	No	No
[58]	Yes	No	No	Yes	No	No
[44]	Yes	No	No	Yes	No	Yes
[75]	No	No	No	No	Yes	Yes
[6]	Yes	Yes	No	No	No	No
[86]	No	No	Yes	Yes	Yes	Yes
[52]	No	No	No	No	Yes	No
[8]	No	Yes	No	No	No	No
[68]	No	Yes	No	No	No	No
[57]	No	No	No	Yes	No	No
[33]	No	Yes	No	No	No	No

TABLE IX. Works Exhibiting Trade offs in Performance Metrics

Reference	Trade off 1	Trade off 2
[22]	Increase in capacity of fog nodes decreases the computing density and hence impacts average latency	Increase of traffic arrival rate increases both traffic load and computing load in network
[54]	Increase in level of heterogeneity decreases maximum link utilization and hence affects bandwidth cost	Increase in level of heterogeneity decreases maximum server utilization but increases average server utilization
[71]	Increase in no of tasks and task size induces delay	As length of task increases energy consumption increases
[7]	Increase in no of gateways reduces energy consumption per node but increases vacation phase	Increase in vacation phase induces delay and hence increased response time for requests
[48]	Increase in fog nodes causes overhead while clustering	Increase in Cell scale (K) and Probe ratio (P) hurts performance due to increased network load

8) *Resource Utilization*: The purpose of this evaluation is to make sure that the cloud system is properly utilizing all its resources. A higher resource utilization will result in lower overall costs as well as reduced energy expenditure and carbon emission.
9) *Degree of Balance*: Load uniformity is a measure of the distribution of workload among the VMs after load balancing. Conversely, degree of imbalance can be regarded as a metric which measures the imbalance among VMs.
10) *Makespan*: It corresponds to the interval of time between the start and finish of a series of tasks or jobs [44].
11) *Cost*: The total cost of providing the service to the user is accounted for by the Cloud service consumer(CSC). A good algorithm for load balancing reduces the service cost for the CSC and increases the CSP's profitability [44].
12) *Energy Consumption*: Massive workflow applications require more data transmission and hence more hardware requirement which increases the demand for energy in terms of execution time and cost. So fog nodes should be efficiently utilized to minimize the energy consumption [43].
13) *Flexibility*: Load balancers should be flexible enough with respect to storage and computation costs making it easier for the users to right-size their resources to their workloads. Admin should be able to configure VMs with any custom number of cores and custom amount of memory. Block volume performance can be reconfigured while they remain online.
14) *Deadline*: The latest time when a service request in the fog system can be completed.
15) *Processing Time*: Describes the length of time it takes for a fog node to complete a service request.
16) *Reliability*: The metric of reliability may include uptime or consistent performance, but it's not just about uptime. The system will automatically transfer the request or job to any other resources (VMs) in case of any system failure or request.

6. Challenges in Fog based Load Balancing

Load balancing has several advantages especially if employed in real-world distributed systems. But, in order to develop load balancers for such systems, several design issues are needed to be addressed. Moreover, the nature of distributed systems and the user requirements pose several additional challenges to any load balancing scheme. The major design issues and challenges encountered while building load balancers are as follows:

1. Estimating the Load: The first thing to consider while building any load balancer is what should be the definition of load at a node in distributed system and what measure should be used for computing it. In order to define a load, there is a need to figure out what is a good indicator of load for the targeted application. The most common definition of load used by many systems is the CPU queue length (length of queue at a node) as it is correlated with

completion time of tasks and is easy to compute. Some of the other definitions of node popularly used are CPU utilization of the node (measured by periodically recording the state of CPU), total number of processes running at the node, resources required by the running processes, etc. However, the stronger the definition of load is (eg. CPU utilization), the larger is the complexity of its computation. Simpler definitions such as number of processes are easier to compute and thus, reduce the overhead.

2. Policies Employed: A load balancing algorithm comprises of 4 major decision making steps —Transfer, Selection, Location and Information Policy. Transfer Policy is responsible for deciding whether a task should be transferred to/from a node (usually based on a threshold). Selection Policy deduces the task which should be transferred among the eligible tasks (usually based on remaining execution time, waiting time, etc.). Location Policy figures out the target node to which the task should be transferred to (usually based on polling, randomly, etc.). Finally, Information Policy decides when and how frequently should the system state collection procedure be triggered, where should the state be collected from and what information needs to be collected. A load balancing algorithm needs to wisely define these policies depending upon its application. Moreover, it also needs to overcome the overhead which some of these policies might offer, a need to collect the information from other nodes as per our location policy which will incur delay.

3. Network Design: While designing a load balancer, there is a need to take care of the kind of network formed by the distributed system. While a network with closely located nodes say, intranet is likely to incur negligible communication delays, a network with spatially distributed nodes might have significant communication delays. Load balancing algorithms must consider potential effects of changes in the network structure. Based on the application, some load balancing algorithms might be tolerant to high delays, but some might have them as the bottleneck.

4. Complexity of the Load Balancing Algorithm: Ideally, load balancing algorithms should be less complex both in terms of their implementation and the computational overhead incurred by their operations. High complexity would also require it to collect more information and would thus, experience higher communication delays. Hence, a highly complex algorithm might effect the overall performance of the system.

5. Meeting Desired Performance Criterion: Different applications may focus on different class of performance measures. Thus, while designing a load balancing algorithm, it might be required to tune it to maximize the targeted performance criterion even if it compromises on some other important measures. Some applications might focus on minimizing the overall completion time of the tasks while some might just be concerned about the overall throughput. Some applications might target to get the same amount of load at each node while some might want to reduce the waiting time of the processes. The desired performance forms a critical aspect of any load balancer.

6. Prioritizing Tasks: Fog computing uses tight delay times to serve a large number of users with a limited number of resources. Thus, tasks should be dealt with according to their priority levels. Priority of a task may depend on various factors like its deadline, the emergency of the request which can be decided based on its source of arrival etc.

7. Task Deadlines: Each task might have a deadline before which it needs to be completed. The load balancer needs to acknowledge this deadline before making any decisions. Transferring the task to another node might reduce the overall completion time of the system but it might imply crossing the deadline of certain tasks. This might not be acceptable in certain scenarios.

8. Heterogeneity: A system might have heterogeneity both with respect to the incoming tasks and the nature of nodes in the system. In a typical distributed system, the incoming tasks have varying resource requirements. Different tasks will thus require different handling by the load balancer. Moreover, each node might have different computation power, storage, memory, etc. The heterogeneous nature of the tasks and the nodes acts as an added challenge to the load balancer as it needs to account for the nature of load.

9. Scalability: The load balancing algorithm should allow the system to scale to user and traffic growth. Normally, in order to scale, a scale-out model (like Google's Maglev) is adopted where more number of nodes are added to the system. The load balancing algorithm needs to quickly adapt to the addition of new nodes or removal of nodes in a large distributed system. It should be effective in balancing load in the scaled system while at the same time being able to make quick decisions with low overhead.

10. Availability & Reliability: A load balancer should be highly available, be quick to response and should be adaptable to faults of nodes involved in load collection and redistribution. In most load balancers, the whole system is controlled by one controller. So, if the controller malfunctions, the entire system would fail. To counter this, some load balancers are often deployed in pairs to avoid single point of failure. However, this would still just entail a 1+1 redundancy. The distributed load balancers do a much better job at handling faults, but they are more complex to design.

11. Migration time: Migration time is the amount of time needed for transferring a task from one system node to another node during load redistribution. This time has to be minimized for better performance.

12. Flexibility & Programmability: A load balancer should be flexible and it should be easy to modify its different components.

13. Upgrading Overhead: A load balancer should be easy to upgrade if needed, say, for incorporating a newly arrived technology.

14. Determinism & Preemption: A load balancing algorithm needs to ensure Determinism, i.e., a task transferred to some other node should output the same result as it would had originally. Moreover, the load balancing algorithm needs to make sure that transferring

a task to a node should not significantly deteriorate the performance of the target node as viewed by the original user of the node. In case of a degraded performance, there should be a mechanism for preempting the transferred task and releasing the resources for running tasks of the node's owner.

7. Case Study: Fog computing in Real time Applications

The term "real-time" refers to applications with low latency that work within a pre-determined timeframe under which users perceive immediate or current events. Most Fog oriented vital time critical applications from various research articles are circumstantiated below.

1) **Smart Factory:** Modern communication methodologies are being transformed by the Industrial Internet of Things (IIoT). In addition to industrial applications like robotics, medical devices and software-defined production processes, IIoT embraces business applications as well. Decentralized, intelligent and efficient solutions are required to guarantee real-time communications, reliability and environmental protection. Many significant works have been published on these applications in [83, 11].
2) **Healthcare:** Health care services have been streamlined through the use of IoT applications assisted by fog computing. Several such works have been carried out by researchers in the field of health care for providing time critical data to the end users that have been discussed in [27, 13, 1].
3) **Smart Grids:** A smart grid is necessary because today's energy demand has outpaced the rate at which energy can be generated traditionally, as well as to curtail or control global warming via reducing gas emissions. Few works that have been carried for smart grid management are [93, 55, 40].
4) **Smart Vehicles:** Introducing mobile cloud computing requires an examination of its agents, which will include cars, robots and humans. These agents have been designed to sense the surroundings, process data and send the results to users' devices. The response in smart vehicles is a crucial aspect which should be addressed immediately as conveyed in [49, 34].
5) **Augmented reality and Real-time Video Analytics based on Fog:** A smartphone, smart glasses or other device with augmented reality applications enables a user to see much more information about an object. Recent projects include Google Glass and Sony SmartEyeglass. These applications require a lot of computing power as well as high bandwidth to process video streaming [67, 3].
6) **Smart Cities:** The concept of smart cities is a key application of IoT that includes everything from smart traffic management to smart energy management. In recent years, *smart cities* has attracted the attention of the science, engineering, and research communities as a solution to urban expansion challenges [46, 12, 90].

8. Conclusion and Future directions

The emerging idea of FC for IoT based applications is rapidly investigated and adopted by researchers, industry and organizations. In a fog system, LB facilitates the distribution of workload on resources equally, to provide services continually and is done by provisioning and de-provisioning application instances along with proper resource utilization. This paper presented an overview of several FC architectures along with their comparison in terms of number of tiers used, technique employed and number of servers/nodes used. A broad classification of existing LB based on policy, load balancer type, system state and migration type are provided. The advantages and limitations of the existing LB strategies based on natural phenomena, agent nodes, application-oriented and network-aware along with general techniques in different computing environments are tabulated. A comparison of recent existing FC based LB mechanisms in terms of pros and cons, tools used and certain tradeoffs with respect to their performance metrics have been tabulated. Further, all possible LB factors and metrics are investigated. Finally, a number of research challenges, case studies in FC environments are presented. This paper focuses on several LB strategies in computing environments that can be mapped or customized to LB in FC environments.

A. *Future directions*

For future implementations, we can consider designing techniques for data replication to manage data in fog computing networks to minimize delay and overall dependencies. Therefore a strategy to improve resource provisioning to address dynamic traffic management is a prevalent issue that needs to be addressed.

Due to the geographic distribution of fog data centers, there are varying levels of CO_2 emissions, which makes optimizing the placement of resources and applications in distributed fog more challenging. Servers that are not needed should not have jobs routed to them as they consume the same amount of power as other nodes and thus should be allowed to enter a power-saving mode. It has related challenges like servers should maintain their state during transition from/to power-saving mode and how many servers to be toggled between these modes as we lack the knowledge of future requests.

Uncertainties in the traffic flow lead to many issues, out of which two prominent issues are: Firstly, performance degradation, where there is a tendency to migrate the same job frequently, and hence it requires a way to tackle the issue of a job being a victim of migration. Secondly, transmission costs are associated with migration. So there is a need to analyze the negative impact of service migration also.

Links or nodes' failures should be tracked and acted upon immediately in fog data centers. When a failure

is observed between the nodes connected directly or via multiple hops, an alternate path should be available in the routing table of the controller node.

References

[1] Mohammad Aazam and Eui-Nam Huh. "E-HAMC: Leveraging Fog computing for emergency alert service". In: *2015 ieee international conference on pervasive computing and communication workshops (percom workshops)*. IEEE. 2015, pp. 518–523.

[2] Mohammad Aazam and Eui-Nam Huh. "Fog computing micro datacenter based dynamic resource estimation and pricing model for IoT". In: *2015 IEEE 29th International Conference on Advanced Information Networking and Applications*. IEEE. 2015, pp. 687–694.

[3] Arif Ahmed et al. "Fog computing applications: Taxonomy and requirements". In: *arXiv preprint arXiv:1907.11621* (2019).

[4] Fayez Alqahtani, Mohammed Amoon, and Aida A Nasr. "Reliable scheduling and load balancing for requests in cloud-fog computing". In: *Peer-to-Peer Networking and Applications* (2021), pp. 1–12.

[5] D Baburao, T Pavankumar, and CSR Prabhu. "Survey on Service Migration, load optimization and Load Balancing in Fog Computing Environment". In: *2019 IEEE 5th International Conference for Convergence in Technology (I2CT)*. IEEE. 2019, pp. 1–5.

[6] Enzo Baccarelli et al. "Learning-in-the-Fog (LiFo): Deep Learning Meets Fog Computing for the Minimum-Energy Distributed Early-Exit of Inference in Delay-Critical IoT Realms". In: *IEEE Access* 9 (2021), pp. 25716–25757.

[7] Fatemeh Banaie et al. "Load-Balancing Algorithm for Multiple Gateways in Fog-Based Internet of Things". In: *IEEE Internet of Things Journal* 7.8 (2020), pp. 7043–7053.

[8] Roberto Beraldi and Hussein Alnuweiri. "Exploiting power-of-choices for load balancing in fog computing". In: *2019 IEEE International Conference on Fog Computing (ICFC)*. IEEE. 2019, pp. 80–86.

[9] Roberto Beraldi et al. "Distributed load balancing for heterogeneous fog computing infrastructures in smart cities". In: *Pervasive and Mobile Computing* (2020), p. 101221.

[10] Flavio Bonomi et al. "Fog computing and its role in the internet of things". In: *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. 2012, pp. 13–16.

[11] Mathias Santos de Brito et al. "Application of the fog computing paradigm to smart factories and cyber-physical systems". In: *transactions on emerging telecommunications technologies* 29.4 (2018), e3184.

[12] Charles C Byers and Patrick Wetterwald. "Fog computing distributing data and intelligence for resiliency and scale necessary for iot: The internet of things (ubiquity symposium)". In: *Ubiquity* 2015.November (2015), pp. 1–12.

[13] Yu Cao et al. "Distributed analytics and edge intelligence: Pervasive health monitoring at the era of fog computing". In: *Proceedings of the 2015 Workshop on Mobile Big Data*. 2015, pp. 43–48.

[14] Ashish Chandak and Niranjan Kumar Ray. "A review of load balancing in fog computing". In: *2019 International Conference on Information Technology (ICIT)*. IEEE. 2019, pp. 460–465.

[15] Sejin Chun et al. "A pub/sub-based fog computing architecture for internet-of-vehicles". In: *2016 IEEE international conference on cloud computing technology and science (CloudCom)*. IEEE. 2016, pp. 90–93.

[16] OpenFog Consortium et al. "IEEE Standard for Adoption of OpenFog Reference Architecture for Fog Computing". In: *IEEE Std* 2018 (1934), pp. 1–176.

[17] Yangyang Dai, Yuansheng Lou, and Xin Lu. "A task scheduling algorithm based on genetic algorithm and ant colony optimization algorithm with multi-QoS constraints in cloud computing". In: *2015 7th international conference on intelligent human-machine systems and cybernetics*. Vol. 2. IEEE. 2015, pp. 428–431.

[18] Kousik Dasgupta et al. "A genetic algorithm (ga) based load balancing strategy for cloud computing". In: *Procedia Technology* 10 (2013), pp. 340–347.

[19] D Chitra Devi and V Rhymend Uthariaraj. "Load balancing in cloud computing environment using improved weighted round robin algorithm for non-preemptive dependent tasks". In: *The scientific world journal* 2016 (2016).

[20] V Divya and R Leena Sri. "ReTra: Reinforcement based Traffic Load Balancer in Fog based Network". In: *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. IEEE. 2019, pp. 1–6.

[21] Shridhar G Domanal and G Ram Mohana Reddy. "Load balancing in cloud environment using a novel hybrid scheduling algorithm". In: *2015 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*. IEEE. 2015, pp. 37–42.

[22] Qiang Fan and Nirwan Ansari. "Towards workload balancing in fog computing empowered IoT". In: *IEEE Transactions on Network Science and Engineering* 7.1 (2018), pp. 253–262.

[23] J Mercy Faustina et al. "Load Balancing in Cloud Environment using Self-Governing Agent". In: *2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA)*. IEEE. 2019, pp. 480–483.

[24] Vangelis Gazis et al. "Components of fog computing in an industrial internet of things context". In: *2015 12th Annual IEEE International Conference on Sensing, Communication, and Networking-Workshops (SECON Workshops)*. IEEE. 2015, pp. 1–6.

[25] Mostafa Ghobaei-Arani, Alireza Souri, and Ali A Rahmanian. "Resource management approaches in

fog computing: a comprehensive review". In: *Journal of Grid Computing* 18.1 (2020), pp. 1–42.

[26] Einollah Jafarnejad Ghomi, Amir Masoud Rahmani, and Nooruldeen Nasih Qader. "Load-balancing algorithms in cloud computing: A survey". In: *Journal of Network and Computer Applications* 88 (2017), pp. 50–71.

[27] Tuan Nguyen Gia et al. "Fog computing in healthcare internet of things: A case study on ecg feature extraction". In: *2015 IEEE international conference on computer and information technology; ubiquitous computing and communications; dependable, autonomic and secure computing; pervasive intelligence and computing*. IEEE. 2015, pp. 356–363.

[28] Diogo Gonçalves et al. "Proactive virtual machine migration in fog environments". In: *2018 IEEE Symposium on Computers and Communications (ISCC)*. IEEE. 2018, pp. 00742–00745.

[29] Jitender Grover and Shivangi Katiyar. "Agent based dynamic load balancing in Cloud Computing". In: *2013 International Conference on Human Computer Interactions (ICHCI)*. IEEE. 2013, pp. 1–6.

[30] Daphné Guibert et al. "CC-fog: Toward content-centric fog networks for E-health". In: *2017 IEEE 19th International Conference on e-Health Networking, Applications and Services (Healthcom)*. IEEE. 2017, pp. 1–5.

[31] Saurabh Gupta et al. "Features exploration of distinct load balancing algorithms in cloud computing environment". In: *International Journal of Advanced Networking and Applications* 11.1 (2019), pp. 4177–4183.

[32] J Octavio Gutierrez-Garcia and Adrian Ramirez-Nafarrate. "Agent-based load balancing in cloud data centers". In: *Cluster Computing* 18.3 (2015), pp. 1041–1062.

[33] Md Sajjad Hossain et al. "Fog Radio Access Networks in Internet of Battlefield Things (IoBT) and Load Balancing Technology". In: *2019 International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE. 2019, pp. 750–754.

[34] Xueshi Hou et al. "Vehicular fog computing: A viewpoint of vehicles as the infrastructures". In: *IEEE Transactions on Vehicular Technology* 65.6 (2016), pp. 3860–3873.

[35] Jasmin James and Bhupendra Verma. "Efficient VM load balancing algorithm for a cloud computing environment". In: *International Journal on Computer Science and Engineering* 4.9 (2012), p. 1658.

[36] Nadeem Javaid et al. "Cloud and fog based integrated environment for load balancing using Cuckoo Levy distribution and flower pollination for smart homes". In: *2019 International Conference on Computer and Information Sciences (ICCIS)*. IEEE. 2019, pp. 1–6.

[37] Sakeena Javaid et al. "Intelligent resource allocation in residential buildings using consumer to fog to cloud based framework". In: *Energies* 12.5 (2019), p. 815.

[38] Anand Jumnal and Dilip Kumar SM. "Energy aware cluster based optimal virtual machine placement in cloud environment". In: *2020 Fourth International Conference on Inventive Systems and Control (ICISC)*. IEEE. 2020, pp. 266–271.

[39] Ahmed Jawad Kadhim and Jaber Ibrahim Naser. "Proactive load balancing mechanism for fog computing supported by parked vehicles in IoV-SDN". In: *China Communications* 18.2 (2021), pp. 271–289.

[40] Muhammad Babar Kamal et al. "Heuristic min-conflicts optimizing technique for load balancing on fog computing". In: *International Conference on Intelligent Networking and Collaborative Systems*. Springer. 2018, pp. 207–219.

[41] Mostafa Haghi Kashani, Ahmad Ahmadzadeh, and Ebrahim Mahdipour. "Load balancing mechanisms in fog computing: A systematic review". In: *arXiv preprint arXiv:2011.14706* (2020).

[42] Mandeep Kaur and Rajni Aron. "A systematic study of load balancing approaches in the fog computing environment". In: *The Journal of Supercomputing* (2021), pp. 1–46.

[43] Mandeep Kaur and Rajni Aron. "Energy-aware load balancing in fog cloud computing". In: *Materials Today: Proceedings* (2020).

[44] Mandeep Kaur and Rajni Aron. "FOCALB: Fog Computing Architecture of Load Balancing for Scientific Workflow Applications". In: *Journal of Grid Computing* 19.4 (2021), pp. 1–22.

[45] Dragi Kimovski et al. "Adaptive nature-inspired fog architecture". In: *2018 IEEE 2nd International Conference on Fog and Edge Computing (ICFEC)*. IEEE. 2018, pp. 1–8.

[46] Rob Kitchin. "The real-time city? Big data and smart urbanism". In: *GeoJournal* 79.1 (2014), pp. 1–14.

[47] Quang Duy La et al. "Enabling intelligence in fog computing to achieve energy and latency reduction". In: *Digital Communications and Networks* 5.1 (2019), pp. 3–9.

[48] Changlong Li et al. "SSLB: self-similarity-based load balancing for large-scale fog computing". In: *Arabian Journal for Science and Engineering* 43.12 (2018), pp. 7487–7498.

[49] Ning Lu et al. "Connected vehicles: Solutions and challenges". In: *IEEE internet of things journal* 1.4 (2014), pp. 289–299.

[50] Mukhtar ME Mahmoud et al. "Towards energy-aware fog-enabled cloud of things for healthcare". In: *Computers & Electrical Engineering* 67 (2018), pp. 58–69.

[51] B Mallikarjuna and P Venkata Krishna. "OLB: a nature inspired approach for load balancing in cloud computing". In: *Cybernetics and Information Technologies* 15.4 (2015), pp. 138–148.

[52] AB Manju and S Sumathy. "Efficient load balancing algorithm for task preprocessing in fog computing

environment". In: *Smart Intelligent Computing and Applications*. Springer, 2019, pp. 291–298.
[53] Vladimir Marbukh. "Towards Fog Network Utility Maximization (FoNUM) for Managing Fog Computing Resources". In: *2019 IEEE International Conference on Fog Computing (ICFC)*. IEEE. 2019, pp. 195–200.
[54] Mirza Mohd Shahriar Maswood et al. "A novel strategy to achieve bandwidth cost reduction and load balancing in a cooperative three-layer fog-cloud computing environment". In: *IEEE Access* 8 (2020), pp. 113737–113750.
[55] Saqib Nazir et al. "Cuckoo optimization algorithm based job scheduling using cloud and fog computing in smart grid". In: *International Conference on Intelligent Networking and Collaborative Systems*. Springer. 2018, pp. 34–46.
[56] Euclides C Pinto Neto, Gustavo Callou, and Fernando Aires. "An algorithm to optimise the load distribution of fog environments". In: *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE. 2017, pp. 1292–1297.
[57] Song Ningning et al. "Fog computing dynamic load balancing mechanism based on graph repartitioning". In: *China Communications* 13.3 (2016), pp. 156–164.
[58] Ryuji Oma et al. "An energy-efficient model for fog computing in the internet of things (IoT)". In: *Internet of Things* 1 (2018), pp. 14–26.
[59] Opeyemi Osanaiye et al. "From cloud to fog computing: A review and a conceptual live VM migration framework". In: *IEEE Access* 5 (2017), pp. 8284–8300.
[60] Gaurang Patel, Rutvik Mehta, and Upendra Bhoi. "Enhanced load balanced min-min algorithm for static meta task scheduling in cloud computing". In: *Procedia Computer Science* 57 (2015), pp. 545–553.
[61] Karan D Patel and Tosal M Bhalodia. "An efficient dynamic load balancing algorithm for virtual machine in cloud computing". In: *2019 International Conference on Intelligent Computing and Control Systems (ICCS)*. IEEE. 2019, pp. 145–150.
[62] Manoj Kumar Patra et al. "A Randomized Algorithm for Load Balancing in Containerized Cloud". In: *2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*. IEEE. 2020, pp. 410–414.
[63] Branko Radojević and Mario Žagar. "Analysis of issues with load balancing algorithms in hosted (cloud) environments". In: *2011 proceedings of the 34th international convention MIPRO*. IEEE. 2011, pp. 416–420.
[64] Neeraj Rathore and Inderveer Chana. "Load balancing and job migration techniques in grid: a survey of recent trends". In: *Wireless personal communications* 79.3 (2014), pp. 2089–2125.
[65] Anees Ur Rehman et al. "Dynamic Energy Efficient Resource Allocation Strategy for Load Balancing in Fog Environment". In: *IEEE Access* 8 (2020), pp. 199829–199839.
[66] Naidila Sadashiv and SM Dilip Kumar. "Cluster, Grid and Cloud Computing". In: *IEEE 6th International Conference on Computer Science & Education (ICCSE 2011)*. 2011, pp. 978–1.
[67] Shaik Mohammed Salman et al. "Fog computing for augmented reality: Trends, challenges and opportunities". In: *2020 IEEE International Conference on Fog Computing (ICFC)*. IEEE. 2020, pp. 56–63.
[68] Bikash Sarma et al. "Fog computing: An enhanced performance analysis emulation framework for IoT with load balancing smart gateway architecture". In: *2019 International Conference on Communication and Electronics Systems (ICCES)*. IEEE. 2019, pp. 1–5.
[69] Kripa Sekaran and KR Kosala Devi. "SIQ algorithm for efficient load balancing in cloud". In: *2017 International Conference on Algorithms, Methodology, Models and Applications in Emerging Technologies (ICAMMAET)*. IEEE. 2017, pp. 1–5.
[70] Muhammad Sohaib Shakir and Abdul Razzaque. "Performance comparison of load balancing algorithms using cloud analyst in cloud computing". In: *2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)*. IEEE. 2017, pp. 509–513.
[71] Shivi Sharma and Hemraj Saini. "A novel four-tier architecture for delay aware scheduling and load balancing in fog environment". In: *Sustainable Computing: Informatics and Systems* 24 (2019), p. 100355.
[72] Anil Singh and Nitin Auluck. "Load balancing aware scheduling algorithms for fog networks". In: *Software: Practice and Experience* 50.11 (2020), pp. 2012–2030.
[73] Gurasis Singh and Kamalpreet Kaur. "An improved weighted least connection scheduling algorithm for load balancing in web cluster systems". In: *International Research Journal of Engineering and Technology (IRJET)* 5.3 (2018), p. 6.
[74] Simar Preet Singh, Anju Sharma, and Rajesh Kumar. "Design and exploration of load balancers for fog computing using fuzzy logic". In: *Simulation Modelling Practice and Theory* 101 (2020), p. 102017.
[75] Fatma M Talaat et al. "A load balancing and optimization strategy (LBOS) using reinforcement learning in fog computing environment". In: *Journal of Ambient Intelligence and Humanized Computing* (2020), pp. 1–16.
[76] Fatma M Talaat et al. "Effective load balancing strategy (ELBS) for real-time fog computing environment using fuzzy and probabilistic neural networks". In: *Journal of Network and Systems Management* 27.4 (2019), pp. 883–929.
[77] Nadim Téllez et al. "A tabu search method for load balancing in fog computing". In: *Int. J. Artif. Intell* 16.2 (2018).

[78] Nguyen B Truong, Gyu Myoung Lee, and Yacine Ghamri-Doudane. "Software defined networking-based vehicular adhoc network with fog computing". In: *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. Ieee. 2015, pp. 1202–1207.

[79] Manisha Verma, Neelam Bhardwaj, and Arun Kumar Yadav. "Real time efficient scheduling algorithm for load balancing in fog computing environment". In: *Int. J. Inf. Technol. Comput. Sci* 8.4 (2016), pp. 1–10.

[80] Manisha Verma and Neelam Bhardwaj Arun Kumar Yadav. "An architecture for load balancing techniques for Fog computing environment". In: *International Journal of Computer Science and Communication* 8.2 (2015), pp. 43–49.

[81] Aarti Vig et al. "Autonomous agent based shortest path load balancing in cloud". In: *2016 8th International Conference on Computational Intelligence and Communication Networks (CICN)*. IEEE. 2016, pp. 33–37.

[82] Violetta N Volkova et al. "Load balancing in cloud computing". In: *2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*. IEEE. 2018, pp. 387–390.

[83] Jiafu Wan et al. "Fog computing for energy-aware load balancing and scheduling in smart factory". In: *IEEE Transactions on Industrial Informatics* 14.10 (2018), pp. 4548–4556.

[84] Shu-Ching Wang et al. "Towards a load balancing in a three-level cloud computing network". In: *2010 3rd international conference on computer science and information technology*. Vol. 1. IEEE. 2010, pp. 108–113.

[85] Tingting Wang et al. "Load balancing task scheduling based on genetic algorithm in cloud computing". In: *2014 IEEE 12th international conference on dependable, autonomic and secure computing*. IEEE. 2014, pp. 146–152.

[86] VM Arul Xavier and S Annadurai. "Chaotic social spider algorithm for load balance aware task scheduling in cloud computing". In: *Cluster Computing* 22.1 (2019), pp. 287–297.

[87] Xiaolong Xu et al. "A heuristic virtual machine scheduling method for load balancing in fog-cloud computing". In: *2018 IEEE 4th International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing,(HPSC) and IEEE International Conference on Intelligent Data and Security (IDS)*. IEEE. 2018, pp. 83–88.

[88] Xiaolong Xu et al. "Dynamic resource allocation for load balancing in fog environment". In: *Wireless Communications and Mobile Computing* 2018 (2018).

[89] Moona Yakhchi et al. "Proposing a load balancing method based on Cuckoo Optimization Algorithm for energy management in cloud computing infrastructures". In: *2015 6th International Conference on Modeling, Simulation, and Applied Optimization (ICMSAO)*. IEEE. 2015, pp. 1–5.

[90] Shanhe Yi et al. "Fog computing: Platform and applications". In: *2015 Third IEEE workshop on hot topics in web systems and technologies (HotWeb)*. IEEE. 2015, pp. 73–78.

[91] Luxiu Yin, Juan Luo, and Haibo Luo. "Tasks scheduling and resource allocation in fog computing based on containers for smart manufacturing". In: *IEEE Transactions on Industrial Informatics* 14.10 (2018), pp. 4712–4721.

[92] Maheen Zahid et al. "Hill climbing load balancing algorithm on fog computing". In: *International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*. Springer. 2018, pp. 238–251.

[93] Muhammad Zakria et al. "Cloud-fog based load balancing using shortest remaining time first optimization". In: *International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*. Springer. 2018, pp. 199–211.

[94] Xiaoqing Zhu et al. "IMPROVING VIDEO PERFORMANCE WITH EDGE SERVERS IN THE FOG COMPUTING ARCHITECTURE." In: *Intel Technology Journal* 19.1 (2015).

Ms. Kavitha M S is currently working as Senior R&D Engineer in Silicon Realization Group at Synopsys, Bangalore. She completed her M.Tech in Computer Science and Engineering from Siddaganga Institute of Technology, Tumkur and B. E from Siddartha Institute of Technology, Tumkur, India. She has four years of Industry experience at Infosys Ltd and Source Code Infotech, Bangalore and nine years of teaching experience at Acharya Institute of Technology, Bangalore.

Dr. Naidila Sadashiv is an Associate Professor in the Department of Computer Science and Engineering at JSS Academy of Technical Education, Bangalore, India. She completed her Ph. D in Computer Science and Engineering from University Visvesvaraya College of Engineering, Bangalore, M.Tech from Dr. Ambedkar Institute of Technology, Bangalore, and B. E from B.V.B.C.E.T, Hubli, India. She is currently teaching Cloud Computing, Unix Operating Systems and her area of research interest includes Cloud Computing and Blockchain. She has published papers in IEEE Conferences and Journals with more than 250 citations. She has won Two Best Paper Awards from IEEE Conferences. She is the recipient of Seed Money to Young Scientists for Research award from Vision Group on Science and Technology, Bangalore. She is a former Assistant Professor at M. S. Ramaiah Institute of Technology, Bangalore. She has total Eighteen years of experience which includes teaching U.G. and P.G. students, research, guiding projects, handling University examinations, organizing seminars, conferences and workshops.

Dr. S M Dilip Kumar received his B. E, M. Tech and Ph. D degrees in 1996, 2001 and 2010 respectively in Computer Science and Engineering discipline. He is currently working as Professor in the Department of Computer Science and Engineering, University Visvesvaraya College of Engineering (UVCE), Bangalore University, Bangalore. Dr. S. M Dilip Kumar is involved in research and teaching B. Tech and M. Tech students of Computer Science and Engineering and has 23 years of teaching experience. He has guided eight Ph. D candidates and six are pursuing Ph. D in Computer Science and Engineering under his guidance in Bangalore University. He has published more than 110 papers in International Journals including Elsevier, Springer, Inderscience and Conferences and has received six best paper awards in International Conferences. He has delivered 40 technical talks in National level seminars, workshops, short-term courses and faculty development programmes. He was the Principal Investigator for a research project in the area of grid computing sponsored by Science and Engineering Research Board, Department of Science and Technology (SERB-DST), Government of India. Another research project sponsored by SERB-DST in the area of Internet of Things is ongoing. He has completed two consultancy projects in the areas of mobile governance and e-FMS sponsored by Government of Karnataka. His current research lies in the area of cloud computing, fog computing and Internet of Things.