symmetry

MDPI

Review

Dynamic Load Balancing Techniques in the IoT: A Review

Dimitris Kanellopoulos [1,*] and Varun Kumar Sharma [2]

1 Department of Mathematics, University of Patras, 26500 Patras, Greece
2 Department of Computer Science and Engineering, The LNM Institute of Information Technology, Jaipur 302031, India
* Correspondence: d_kan2006@yahoo.gr

Abstract: The Internet of things (IoT) extends the Internet space by allowing smart things to sense and/or interact with the physical environment and communicate with other physical objects (or things) around us. In IoT, sensors, actuators, smart devices, cameras, protocols, and cloud services are used to support many intelligent applications such as environmental monitoring, traffic monitoring, remote monitoring of patients, security surveillance, and smart home automation. To optimize the usage of an IoT network, certain challenges must be addressed such as energy constraints, scalability, reliability, heterogeneity, security, privacy, routing, quality of service (QoS), and congestion. To avoid congestion in IoT, efficient load balancing (LB) is needed for distributing traffic loads among different routes. To this end, this survey presents the IoT architectures and the networking paradigms (i.e., edge–fog–cloud paradigms) adopted in these architectures. Then, it analyzes and compares previous related surveys on LB in the IoT. It reviews and classifies dynamic LB techniques in the IoT for cloud and edge/fog networks. Lastly, it presents some lessons learned and open research issues.

Keywords: IoT; load balancing; resource allocation; workload management; optimization

check for updates

Citation: Kanellopoulos, D.; Sharma, V.K. Dynamic Load Balancing Techniques in the IoT: A Review. *Symmetry* **2022**, *14*, 2554. https://doi.org/10.3390/sym14122554

Academic Editors: Yang (Cindy) Yi and Zhen Zhou

Received: 29 October 2022
Accepted: 29 November 2022
Published: 2 December 2022

1. Introduction

More and more low-cost and tiny network devices are incessantly being connected to the Internet. Excluding conventional machines, billions of smart 'things' interact and communicate without any human interference and form the Internet of things (IoT) [1–3]. Smart things (objects) are constrained devices such as low-power wireless sensor nodes with minimal processing and memory capacities. In IoT, end-users may ignore the available resources, services, and capabilities. The IoT brings plentiful benefits to human life through the environment where intelligent services are provided to utilize every activity anytime and anywhere. Typically, the low-power wireless sensor devices in IoT acquire the physical environment information (e.g., temperature, pressure, and motion) to provide intelligent services. IoT applications are monitoring and, consequently, making immediate decisions for efficient management. The things are also reporting their conditions, such as their battery status and fault reporting for predictive maintenance. It is noteworthy that the above environmental information is scalar and periodic, requiring less memory and fewer computational resources.

Acquiring multimedia content such as video from the physical environment leads to a specialized subset of IoT, referred to as the 'Internet of multimedia things' (IoMT) [4,5]. Video in IoMT is massive, and its transmission is more bandwidth-hungry than the conventional scalar data traffic in IoT. Higher processing and memory resources are required for real-time communication. Moreover, the delivery of multimedia data should be within the bounds of quality-of-service (QoS) constraints (i.e., delay and jitter). Multimedia devices require additional functionalities such as higher processing and memory resources to process the sensed multimedia information. The inherited characteristics of multimedia information impose several restrictions on the design of IoT. For example, certain QoS requirements such as end-to-end delay, jitter, and error rate must be bounded to ensure acceptable delivery of multimedia content. IoMT applications also pose a new set of strict requirements on video codecs. An IoT device has limited energy and processing capability. Thus, it is desirable to have a video codec with low-complexity encoding. As a result, the complexity of encoding must be shifted to the cloud [4]. In addition, resilience to transmission errors, high data rate with low power, and delay bound for video streaming are preferable [5]. Integrating multimedia content in IoMT leads to narrative applications [6,7]. Typical examples are intelligent multimedia surveillance systems for habitat monitoring, traffic monitoring systems for road management, multimedia-based industrial monitoring systems, and remote multimedia-based monitoring of an ecological system. IoT covers wide-ranging application fields such as healthcare [8], industrial [9], smart city [10], environmental [10], infrastructural, and commercial [11].

The usage of an IoT network can be optimized when well-defined problems are addressed, such as scalability, heterogeneity, reliability, security, privacy, routing, energy constraints, QoS metrics, and congestion [4]. These problems must be addressed when IoT applications are deployed into real-time scenarios.

Load balancing (LB) [12] plays a crucial role in IoT as it allocates suitable resources to user tasks for optimizing the use of resources. In an IoT network, there are two types of resources: (1) nodes' (things') resources, including the computational resources, storage capacity, and energy resources, and (2) network resources, including bandwidth, load balancer, and traffic analyzer. An efficient LB technique prevents overload and improves the performance of a large-scale IoT network [12]. It improves the QoS metrics, including response time, cost, throughput, performance, and resource utilization. IoT devices are generating many events that may cause heavy traffic (bottleneck) on some data paths. Imbalanced traffic load across the network causes high latency (delay) in some routes and loss of data packets, as well as decreases the packet delivery ratio (PDR). To avoid congestion in IoT, an efficient LB method can distribute loads among different routes. Such load distribution is accomplished using local network information, such as the network topology. An LB mechanism distributes the load among different resources. Thus, optimal network resource utilization is feasible while the performance of IoT applications can be improved. An efficient LB technique can reduce the response time [13], overload [12], packet loss ratio (PLR), etc. PLR indicates the percentage of data packets lost with respect to data packets sent [14]. An efficient LB scheme can increase scalability [15], reliability, and network lifetime. Scalability shows how well an IoT system can perform the LB procedure with a limited number of hosts or servers [16]. Reliability shows how well an IoT network performs its required requests in a defined time if some host failure happens [17]. In addition, the network lifetime can be prolonged if energy consumption is minimized. To this end, an efficient LB technique can improve energy consumption [18,19].

An LB technique can be implemented either in physical devices or software. It can be *centralized* or *distributed*. In a centralized technique, a *central control node* (object) monitors the IoT network and makes accurate decisions regarding LB. For this reason, this node stores the knowledge based on the entire network. If a failure occurs, this node must be repaired quickly. In a distributed technique, *every node* transfers the load to adjacent underloaded nodes. Each node must maintain a 'local knowledge base' to guarantee the efficient distribution of loads. Moreover, it makes a decision about the distribution of load based on its own observed information about the system. If all nodes in the system cooperate to attain a common goal or decision making, then it is called a cooperative LB; otherwise, it is a non-cooperative LB. An LB technique can be static, dynamic, or both. In a *static* balancing technique, the current state of the system is not considered during decision making. Thus, the user's behavior is not predictable. The rule (for LB) is programmed in the *load balancer* by only considering preliminary information about the IoT system. Static techniques are inappropriate for distributed systems that change state dynamically. These techniques only work well if lower load fluctuation occurs in the nodes. In contrast, a dynamic LB technique is flexible and improves system performance. It dynamically distributes the load by taking into account the pattern that is programmed in the load balancer. As a result, dynamic techniques are better than static techniques.

More effective LB algorithms for IoT must be designed in the future. This survey paper focuses on dynamic LB techniques for IoT. This survey does not provide a systematic literature review (SLR) methodology as it covers many different viewpoints of LB. However, many different research questions have been proposed in dealing with the critical issues of LB for IoT. We selected the latest published LB techniques by searching popular academic databases such as IEEE explorer, Science Direct, SAGE, Google Scholar, ACM, Wiley, Emerald, and Springer. To the best of our knowledge, this is the first survey that comprehensively reviews the latest LB schemes for IoT considering cutting-edge computing and networking paradigms (cloud, edge/fog networks) adopted in IoT. It also presents LB from different viewpoints. For example, it presents LB solutions for multipath communication in IoT, while it discusses LB problems that emerge in an MQTT cluster-based environment. The Message Queuing Telemetry Transport (MQTT) standard is used for IoT messaging while a load balancer is set up in front of a cluster of MQTT brokers to distribute MQTT connections and device traffic among clusters of MQTT brokers. The paper aims to survey the existing techniques, describe their properties, and clarify their pros and cons. The paper's contributions are as follows:

1. It considers the recent LB techniques in IoT.
2. It provides a classification of LB mechanisms.
3. It presents the advantages and disadvantages of the LB algorithms in each class.
4. It outlines future research directions to improve the LB algorithms.

The remainder of this paper is structured as follows: Section 2 discusses well-known IoT architectures in which LB methods have been proposed; Section 3 presents the leading computing and networking paradigms adopted in IoT environments; Section 4 presents related surveys on LB techniques in IoT; Section 5 presents recent LB techniques in IoT, while Section 6 presents some lessons learned; Section 7 provides future research directions; lastly, Section 8 concludes the paper.

2. IoT Architectures

The deployment of IoT applications is associated with the development of a standard reference architecture that refers to the significant IoT features and supports probable future extensions. A classic IoT architecture must be well-defined, designed, scalable, interoperable, backward-compatible, and secure, making the exclusive IoT perception deployable in the real sense [1]. As per the current scenario, commercial or cost-effective architecture still needs to be standardized. IoT devices can be [3] resource-well-equipped (Rwe) or resource-constrained (Rc) devices. Rwe devices have the software and hardware competence to support the Transmission Control Protocol/Internet Protocol (TCP/IP) protocol suite. IoT applications control the IoT devices that support the TCP/IP protocol suite. Such applications have been implemented on top of frameworks and application protocols comprising Constrained Application Protocol (CoAP) [20], Representational State Transfer (REST) [20], Advanced Message Queuing Protocol (AMQP) [21], and others. The devices that do not support the TCP/IP protocol suite cannot interoperate straightforwardly with the Rwe devices. This interoperability problem limits the potential of IoT applications. For this problem, numerous efforts [22–26] have been made in the literature.

An IoT-based communication system should be competent to interconnect millions of homogeneous and heterogeneous devices via the Internet. Hence, there is a dire necessity for a simple layered architecture [3].

Figure 1 shows three well-established IoT architectures [3]: (a) the three-layer IoT architecture, (b) the middleware-based IoT architecture, and (c) the service-oriented architecture (SOA)-based IoT architecture and the five-layer IoT architecture. The three-layer IoT architecture [27] is the most elementary model among the proposed architectures. Other architectures complement extra notions of the IoT models [3].

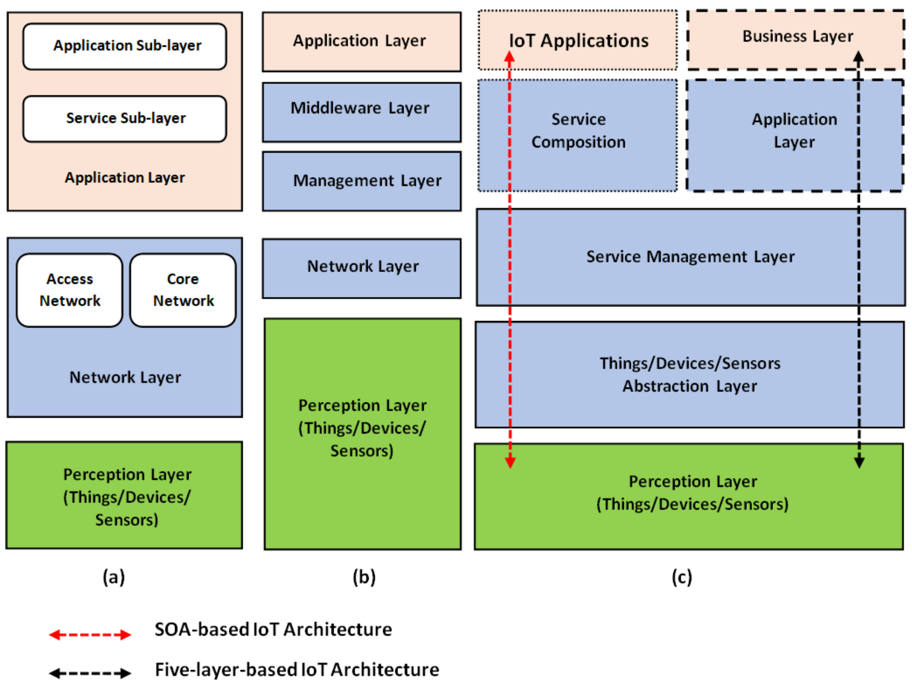

Figure 1. IoT architectures: (**a**) the three-tier IoT architecture; (**b**) the middleware-based IoT architecture; (**c**) the service-oriented architecture (SOA)-based IoT architecture and the five-tier IoT architecture.

2.1. The Three-Tier IoT Architecture

The key model [27] of the three-tier IoT architecture (Figure 1a) includes the perception layer, the network layer, and the application layer.

– *Perception Layer*: This layer contains sensing hardware, scalar or multimedia sensors, and actuators. These devices sense, acquire, and preprocess data from the physical environment. They often send the environmental data to the centralized servers through gateways using various access network technologies. A variety of environmental sensors can be deployed in the monitoring area. Sensors create a network topology in the structure of self-organizing and multiple hops. This network system contains sensor nodes, sink nodes, and management nodes that perform monitoring tasks (initiated by the end-users). The captured data are transmitted through sink nodes by multi-hop. However, the network topology often changes because a few nodes are more prone to failure due to energy consumption and environmental impact. The needless links must be detached by energy control and backbone node selection to achieve an efficient network topology for data forwarding and, thus, ensure network connectivity and coverage.
– *Network Layer*: This layer constructs an efficient network topology for data forwarding. It decides on the power-efficient optimum route to transmit the data to the IoT servers, devices, and applications via the Internet. Various networks, such as WiFi, Ethernet, 3G, Long-Term Evolution (LTE), and 5G, can be used. The Access Network sublayer interconnects various devices and applications through interfaces or gateways using numerous communication protocols. The networking models (included in this layer) can provide high data transmission capacity for nodes. For example, they can transmit the data to the cloud server through sink nodes, super nodes, and other relay units. High data transmission capacity is often required to forward the big data to cloud servers. In addition, self-organizing routing protocols can be used to enhance the robustness of networking models.
– *Application Layer*: This layer contains various IoT applications, such as WiFi applications, wireless sensor network (WSN) applications, and vehicular network applications. WiFi networks support different protocols and have been widely adopted

in homes, cities, and healthcare systems. Users can control the smart devices which connect to WiFi networks. A vehicular network application can monitor emergency traffic events and make traffic predictions on the basis of real-time traffic data. A WSN application can monitor environmental data such as humidity and temperature. The application layer consists of two sublayers: (1) the service sublayer, and (2) the application sublayer. The service sublayer provides data analytics, information management, data mining, and decision-making services. The application sublayer provides the required IoT services to the end users or machines. It delivers the services demanded by clients. Furthermore, this layer can offer decent QoS and quality of experience (QoE) to satisfy the client's needs.

2.2. Middleware-Based IoT Architecture

A middleware-based IoT architecture [28] (shown in Figure 1b) has two extra layers: (1) the middleware layer, and (2) the management layer. Middleware can shield the differences between different operating systems (e.g., TinyOS, Contiki, and LiteOS) in IoT and different network protocols to provide a high QoS for different applications. The management layer coordinates various functionalities (subsystems) such as query manager, storage, event processing, and networking. Notably, most of the popular middleware services use a proprietary protocol, which is difficult to achieve interoperability. Furthermore, middleware services have a time delay and memory overhead due to the incompatible protocols of subsystems. However, one advantage of a middleware-based architecture is that the concepts of the IoT can be integrated with autonomous control by applying deep reinforcement learning (DRL) in the middleware layer. Thus, IoT systems can provide a dynamic and interactive environment. Typically, a software agent (i.e., an IoT device) can be trained to behave smartly. For example, it can (1) sense the environment's state (e.g., home temperature), (2) perform actions, such as turning HVAC on or off, (3) and learn through the maximizing accumulated rewards it receives in the long term.

2.3. The 5-Tier IoT Architecture

The 5-tier IoT model (Figure 1c) is often considered with the service-oriented architecture (SOA) (Figure 1c). In such SOA architecture, the services use protocols that illustrate how they pass and parse messages through description metadata (i.e., the functional and QoS characteristics of the service). The 5-layer IoT model includes the following layers: (1) the perception (object) layer, (2) the things (objects) abstraction layer, (3) the service management layer, (4) the application layer, and (5) the business layer. In this model, the perception layer transfers data to the things abstraction layer via safe channels [3].

– *Things Abstraction Layer*: This layer transfers data generated from the perception layer to the service management layer via some safe channels. Such data transfer can be achieved via technologies such as 3G/4G, GSM, WiFi, and RFID. Many advanced computing paradigms and functionalities, such as cloud computing (CC) and edge computing (EC), and data managing procedures can be implemented in the things abstraction layer [3]. For example, the CC paradigm can be implemented in this layer to enable large-scale IoT systems to handle massive amounts of data with increasing heterogeneity levels [29]. Cloud servers have powerful analytical computing capacity and data storage capabilities. They can also make decisions on the basis of analytical results. Smarter decision making is feasible using effective cloud computing. A cloud server can flawlessly implement communication for heterogeneous systems. Compared to middleware, CC has better heterogeneity capacity in IoT because of its powerful data analytical feature. Fog computing (or 'clouds at the edge') is a technology that allocates services near the devices to improve the QoS. It is a geographically distributed paradigm that complements CC to provide services. Fog computing (FC) can also be incorporated into the things abstraction layer. The fog system can extend storing and computing to the edge of the network. This can solve the difficulty of service computing in delay-sensitive IoT applications and enable location awareness as well as mobility support [30]. FC operates on 'instant data', i.e., real-time data generated by sensors or users. Generally, IoT intelligence can be offered at three levels: (1) in CC infrastructures, (2) in edge/fog nodes, and (3) in IoT software-defined networking (SDN) devices [31]. The need for intelligent control and decision at each level depends on the time sensitivity of the IoT application. For instance, a camera on an autonomous car must detect obstacles in real-time to prevent accidents. This quick decision making is impossible by transferring data instances from the vehicle to the cloud and returning the predictions back to the vehicle. Instead, all the operations should be performed locally in the vehicle. Such a real-time IoT scenario cannot be implemented within a cloud-based IoT environment.

– *The Service Management Layer*: This layer is mainly accountable for order handling, grievance handling, and billing. Furthermore, this layer is responsible for providing interaction among services and service providers [32].
– *The Business Layer*: This layer suggests the management of IoT services and system actions. Furthermore, this layer can assist in building charts, graphs, models, etc., using the incoming data from the Application Layer [3].

3. Computing and Networking Paradigms Adopted in IoT Environments

IoT devices demand the handling of massive generated data, efficient networking, and intelligent management of storage and computing resources. Recent computing paradigms strive to satisfy these requirements to provide resource-effective and timely services. The evolution of these paradigms has gone through the phases of the single computing machine, a cluster of computing machines (cluster computing), network computing, and the CC paradigm.

3.1. The Cloud Computing Paradigm

CC providers offer cloud services to IoT clients/users on a pay-per-use basis anytime, anywhere. CC [33] provides centralized storage and computation facilities. It allows the distribution of resources (servers, storage, network, etc.) to perform user tasks in dedicated data centers. In this way, faster service is provided to IoT clients. This activity requires excellent control and management of user workloads and resources. CC works sufficiently for non-real-time IoT applications running over conventional IP-based networks. However, delay-sensitive IoT applications impose delay requirements, and CC could no longer satisfy the needs of such applications. The CC paradigm unsuccessfully struggled to solve the problem of the wide area network (WAN)'s latency and its limited traffic transfer capacity. In IoT, this traffic transfer capacity depends on the huge amount of generated IoT data transmitted on it. Figure 2 shows the CC paradigm architecture that offers two viewpoints, i.e., (i) front-end and (ii) back-end.

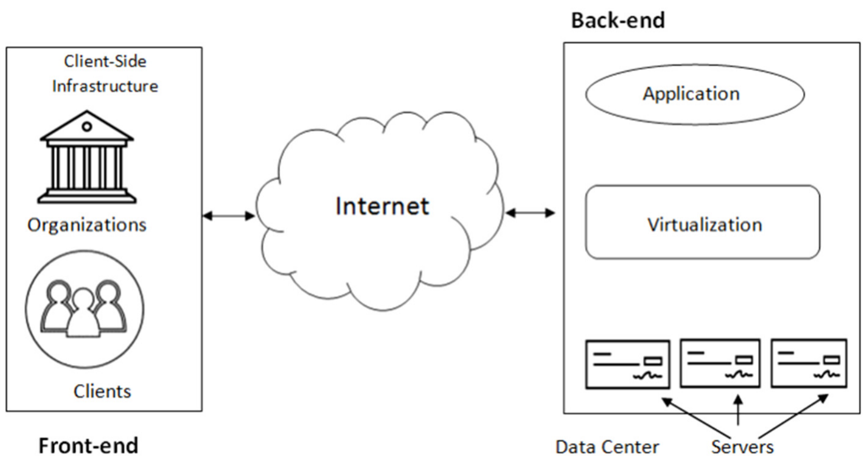

Figure 2. The CC architecture, adapted from [34].

The first (i) viewpoint includes the clients/consumers/users and is utilized by consumers. This comprises various applications and user-level interfaces through which users can access the CC-based platform [35]. The second viewpoint (ii) includes the service suppliers who utilize the back-end support and manage all the necessary resources to provide CC services. The second viewpoint (ii) comprises massive storage capability, deploying models, virtual machines (VMs), physical machines (servers), and traffic monitoring/control schemes. The clients' requests are obtained from the IoT application, and the required resources are allocated through the notion of *virtualization*. Virtualization allows some VMs to be created on a single physical machine. These VMs are independent and have different configurations. Virtualization assists in managing the allocation/utilization of resources and scheduling in the cloud and maintaining the workload balance in the complete system [34].

Initially, three service models were deployed for the cloud: (1) software as a service (SaaS), (2) platform as a service (PaaS), and (3) infrastructure as a service (IaaS) [36]. Notably, IaaS offers excellent QoS to consumers.

LB in a Cloud-Based Network Environment

A client sends its request through the Internet, and all requests come in the manner of VMs. The IaaS service model assists clients by providing a virtualization platform by extensively constructing VMs. These VMs support clients in completing their assignments within an acceptable deadline. Subsequently, computing resources must be efficiently managed and utilized. Furthermore, the workload scheduling policies must be competent enough to schedule the tasks judiciously, reducing the makespan [36].

Figure 3 shows the common planning for workload scheduling via VMs in the CC paradigm. W_1, W_2, … , W_N are the independent workloads that need to be scheduled. For instance, this cloud environment is well-supported by the deployed physical servers (PSs) (i.e., PS_1, PS_2, … , PS_N). On top of PSs, VMs are hosted and maintained through VM monitoring. The resource managers (RMs) are deployed and executed in PSs that monitor the availability (and nonavailability) of PS resources. At point A (Figure 3), the workload scheduler receives incoming requests from clients/users. At point B, the workload scheduler accumulates all the useful resource utilization information from an already deployed RM module to get and sustain the environment's precise or up-to-date current status. Then, the workload scheduler schedules the requested assignment on an appropriate VM on the basis of the accumulated information about the request (from the service supplier) and feedback data from RMs (point C)). Lastly, the workload scheduler assigns the selected VMs to PSs (point D) [37].

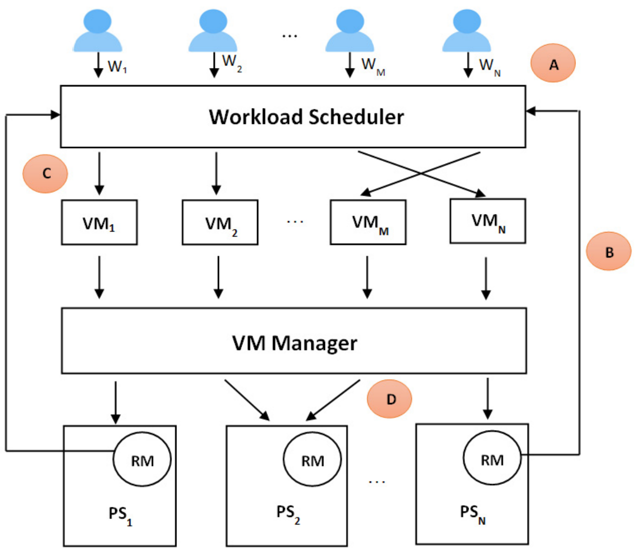

Figure 3. Workload scheduling via VMs in the CC paradigm, adapted from [37].

Two problems are actively being considered: (1) LB or efficient load scheduling among VMs, and (2) reducing the makespan of the workloads. Workload scheduling struggles to maintain an appropriate balance of load amongst VMs, which further assists in reducing the completion time of a workload. Nonetheless, inappropriate workload scheduling leads to the problem of an unbalanced load on the VMs, which ultimately increases the completion time of a workload and thus reduces the system's performance.

3.2. The SDN Paradigm

SDN enables the softwarization of networks by separating the data and control planes [38]. Figure 4 shows SDN's general architecture having three planes: application plane, control plane, and data plane. This model displays how the networking components in the data plane interact with those in the control plane via the southbound application programming interface (SB-API) or the SB-API's protocol (i.e., OpenFlow [39]). The data plane consists of physical networking components which act as forwarding elements. It makes routing commands and information transfer more efficient. The control plane includes at least a single SDN controller, which acts as a centralized entity for routing packets. Another interface, northbound API (NB-API), acts as an intermediate between the application and control planes. SDN applications are located and implemented inside the application plane [39–42]. By utilizing the OpenFlow protocol, SDN allows network policy designers and administrators to get a global network topological view which assists them in dynamically monitoring, adapting employed policies, and reconfiguring the network settings. Moreover, the OpenFlow protocol assists in managing the communication between the SDN central controller and other network components (i.e., OpenFlow switches).

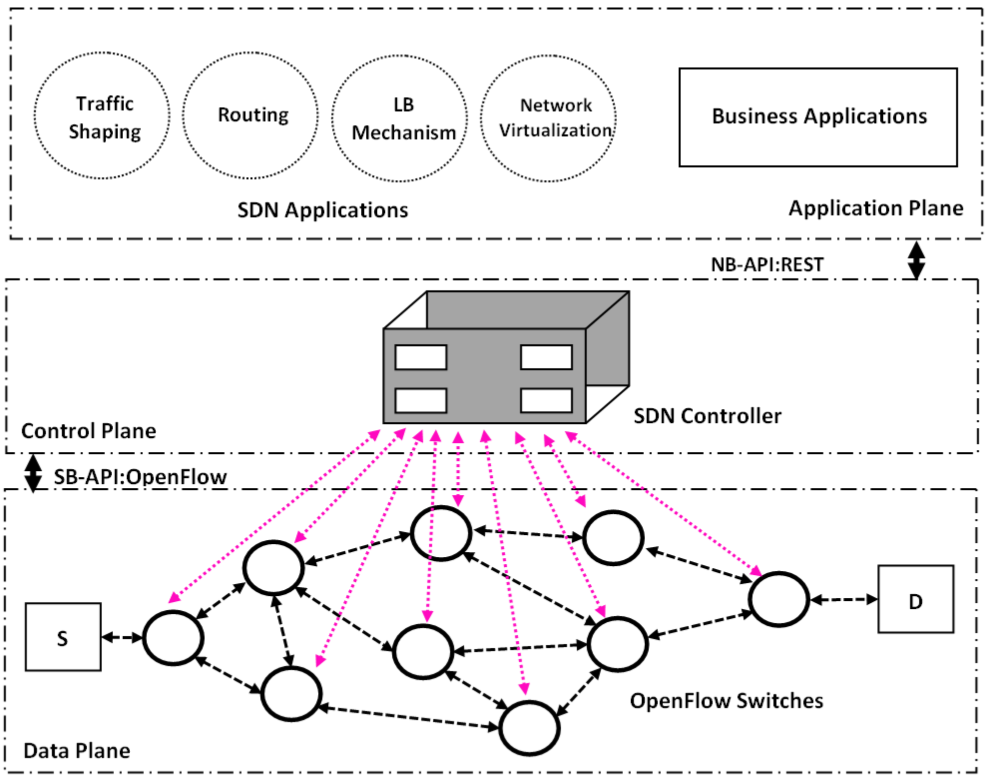

Figure 4. General architecture of SDN, adapted from [39–43]. S: Source, D: Destination.

LB in the Software-Defined IoT

Conventional IP-based networks are highly restrictive while a little flexibility is provided to network designers for implementing a new LB policy. These networks are based on static routing policies and use link state routing protocols, such as the intermediate system to intermediate system (IS–IS) and open shortest path first (OSPF), which attempt to obtain link quality dynamically. Initially, the idea of routing through the shortest path was perfect. As the IP-based network evolved and stricter user requirements emerged, this impression collapsed. The usage of shortest path routing (i.e., static routing) leads to the problem of multiple bottleneck creations, which degrades the network performance altogether. Various load distribution policies [44–47] were automatically adapted on the basis of various link quality metrics while relying on static routing. However, in intelligent IoT networks, effective LB is needed. All the load distribution policies are inefficient as they are designed for the traditional IP-based network only. Moreover, these policies failed in the conventional IP-based network themselves. The reason is simple: the routers often receive packets (traffic) at any time, and their receiving times can significantly vary. Subsequently, this generates an inconsistent global view of the current network state from the viewpoint of all the available routers in the system. It is very problematic to acquire the global view of the IoT network in conventional IP-based networks. It is impossible to estimate the optimality of the current version of the LB mechanism implemented in such networks [43].

Conventional LB techniques are hard to change due to their nonprogrammable design implemented by the vendors. SDN-based load balancers are programmable and offer a platform to the protocol developers on which they can implement their LB schemes. Further, there is no requirement for additional special hardware in SDN which can act as a load balancer. Instead of relying upon a traditional routing protocol, an SDN controller (using the OpenFlow protocol) with an already in-built LB mechanism takes data path control decisions [43]. An SDN controller can utilize network resources efficiently, reduce the network overhead smartly, and eventually get minimum response time by efficiently dispensing the load on IoT devices. An SDN controller can collect information from the upper layers (networking protocols) along with the lower layers/sublayers (critical knowledge of the Physical and Media Access Control (MAC) layers)) together with the current network infrastructure conditions to make an intelligent routing decision. The gathered dynamic information, such as the current status of network links, the average number of packet retransmissions, and the congestion information at routers can assist in improving LB decisions. An SDN controller can have a global view of the network topology and its parameters through which it can decide about LB. Therefore, it can dynamically determine and assess all the feasible flow (data) routes between each pair of nodes in the network by updating the global network topological information. Furthermore, this information assists in making intelligent routing decisions by assessing every available global route's traffic/load situation.

3.3. The Edge Computing Paradigm

EC brings storage and computational resources nearer to the data sources. Thus, the response time is improved. At the same time, EC assists in reducing avoidable dependence on unreliable Internet connections [48–51]. EC can act as an alternate for the CC paradigm for IoT applications that necessitate high accessibility and speed. This happens because IoT applications suffer from the issues of unreliable Internet connectivity somewhere or the other. Whenever the Internet speed goes down (or any connectivity failure occurs), this leads to higher response times, reducing the application's performance as the application availability is decreased. For this reason, EC eliminates or minimizes this Internet dependency by locating the computation and storage services nearer to where it is required. This concept improves the response time (or speeds up the application's response time).

Figure 5 displays the architecture of EC. It comprises three planes: Device plane, EC plane, and Cloud infrastructure plane. The device plane contains intelligent IoT devices such as tablets, smartphones, actuators, and sensors that aim to accumulate and communicate data. In the device plane, the sensing competence of these smart devices is prioritized, regardless of their computing capability. Moreover, the EC plane has distributed edge devices (nodes) that act as an interface between the cloud infrastructure and device plane. These edge devices may act as smart devices themselves or can utilize for making connections (i.e., gateways and routers). The cloud infrastructure plane receives the data conveyed by the EC plane, which stores it for a longer time in those clouds. The storage and logical processing, which may not be accomplished via the local EC plane, must only be performed in clouds. Furthermore, the cloud infrastructure plane can robustly regulate the scheme of EC plane concerning network resources' distribution [52]. The computing competence-enabled edge devices and nodes execute a large amount of computing workload such as data processing, device supervision, provisionally storing, and decision building to ultimately reduce Internet dependence.

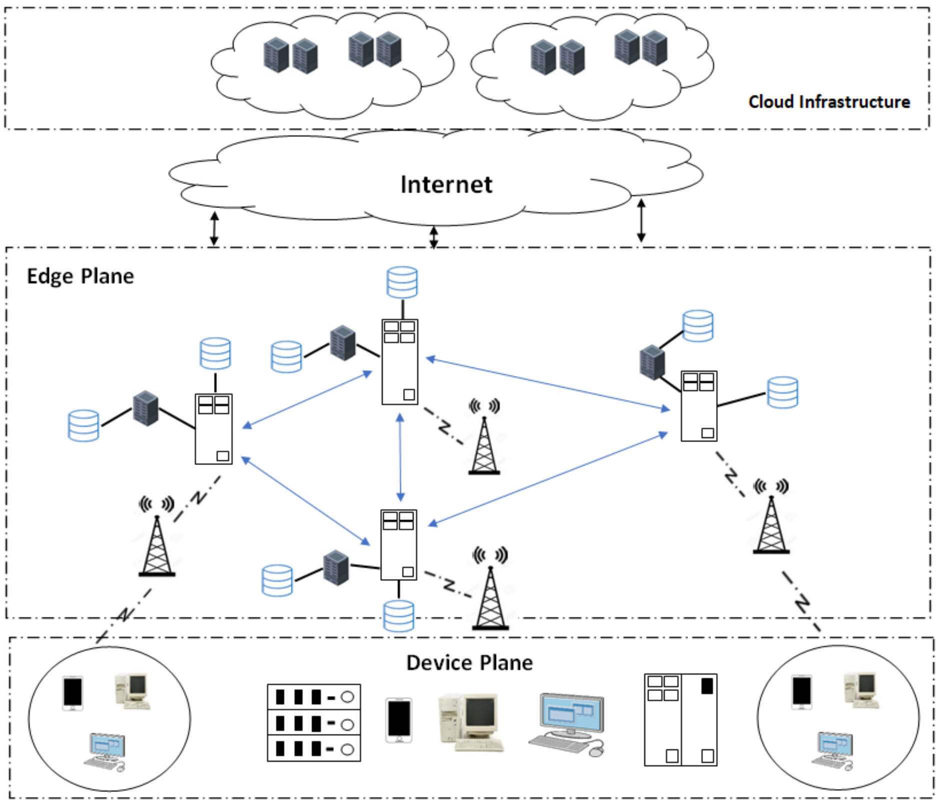

Figure 5. General architecture of EC, adapted from [52].

3.4. The Fog Computing Paradigm

Fog computing (FC) is an extended paradigm initially motivated by the CC paradigm from moving the core to the edge of the network. FC brings computational resources closer to the IoT/device plane so that the main computation can be achieved locally. This leads to the advantage of having lower network latency and faster application availability. However, there is a requirement with FC; it cannot work standalone without a cloud system. FC follows the concept of multitier architecture which provides great flexibility to the system [53–55].

Figure 6 depicts the architecture of FC, which comprises three planes: IoT/device plane, fog infrastructure plane, and cloud infrastructure plane. The fog infrastructure plane includes single or multiple fog domains with compound heterogeneous fog devices/nodes such as edge gateways, switches, access points, PCs, routers, tablets, smartwatches, smartphones, and set-top boxes. The IoT/device plane consists of IoT and end-user devices. Furthermore, the fog infrastructure and the IoT/device plane communicate through a local area network (LAN). At the same time, the IoT/device plane can be connected to the cloud infrastructure plane via a WAN [53].

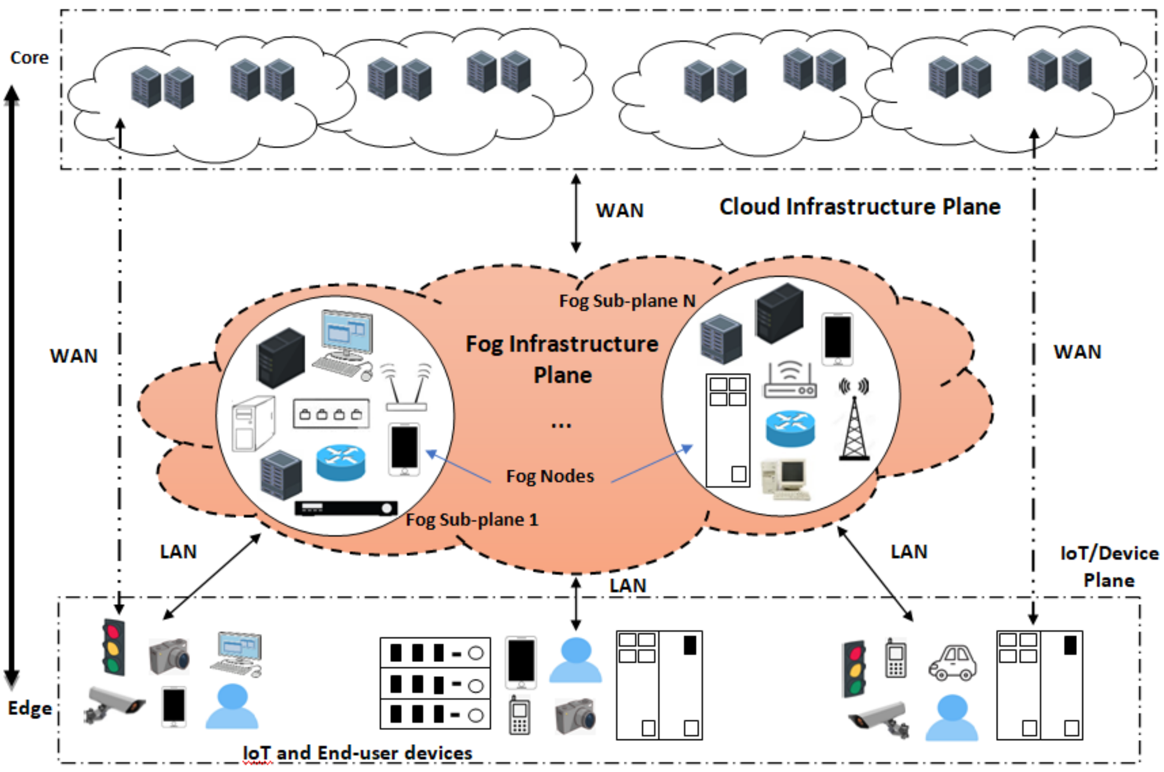

Figure 6. General architecture of Fog computing, adapted from [53].

Task Offloading in the Fog and EC Paradigms

In a fog architecture (Figure 6), the sensed data in the IoT/device plane through IoT and end-user devices are communicated to the fog domain(s) (specifically to fog nodes). As soon as the volume of sensed data increases, many of the available fog nodes in fog domain(s) are overburdened. Subsequently, this leads to the problem of higher response and delivery time due to a rapid increase in data processing time caused by an increase in computational load at fog nodes [56]. To avoid these issues, proper coordination or synchronization between the fog nodes is required. An appropriate task offloading policy is needed amongst these nodes to dynamically transfer the load from an overloaded fog node to the underloaded nodes. This causes appropriate resource utilization, proper resource availability balance amongst fog nodes, and smaller power consumption.

3.5. Performance Evaluation Metrics for LB Schemes

This section describes the performance metrics adopted to evaluate LB schemes implemented on the computing paradigms. The selection of the performance evaluation metrics depends on the computing paradigm wherein an IoT application and LB scheme are deployed. As FC and EC inherit the features of the CC paradigm, common performance evaluation metrics can be adopted in the context of these paradigms. However, the evaluation of some metrics can be very complicated in some contexts. For example, when considering the response time in the cloud, its evaluation is a complex procedure for a policy designer [57].

Table 1 presents the fundamental performance evaluation metrics for LB.

Table 1. Performance evaluation metrics for LB.

Metric	Description
Throughput	It states the number of workloads served when LB obtains efficiency in a unit of time.
Response time	It is the aggregate period it takes for a user to get responses from the IoT application. It comprises the transmission, propagation, processing at middle-boxes (waiting), and service time. It mainly depends on the available bandwidth, the number of users contending for resources simultaneously, and processing time. A larger number of workloads (per unit time) must be served to obtain a faster response time.
Scalability	The LB algorithm must be adaptable (scalable) when the number of receiving requests is increasing. The algorithm should be scalable enough to handle the increasing load and workload requests during its lifetime.
Fault tolerance	Any element can fail when the IoT system does its tasks flawlessly if a VM fails (i.e., low or no available resources) while performing a task. In that case, there must be some way so that the task has to be offloaded from the overburdened VM to a lesser-loaded VM, which ultimately executes the job completely. The failure element(s) perspective can assess the degree of fault tolerance.
Migration time	It is an aggregate period required to transfer a workload from an overloaded VM to an underloaded VM. This offloading procedure should happen without influencing the system's or application's availability. An efficient LB policy has low migration times.
Resource utilization	It is the degree of resource (i.e., CPU and memory) usage in the computing system. Due to the incoming workload requests, the demands for resources have increased. Thus, the policy must utilize the available resources efficiently to accommodate almost all incoming requests. In a decent LB policy, resource utilization is as maximum as possible.
Overhead	It describes the extra operational cost after deploying an LB scheme. This overhead can be due to the huge transmission of control packets (i.e., communication overhead [19,58–60], massive migrations, or high offloading amongst VMs). Any extra resource usage in handling and completing requests is treated as an overhead. There are other types of overhead that influence the overall performance of an LB scheme, such as flow stealing [61], flow statistics collection [62], and synchronization overhead [63].
Makespan	It is an aggregate period taken to complete an assigned workload and designate required resources to the consumers in the system. A shorter time leads to the consideration of a better LB policy.
Power consumption/management	It is the power consumed by each node while maintaining the system's connectivity, task assignment, task migration/offloading, and complete execution [64]. It does not matter whether the task is fully accomplished or not; the energy is consumed somewhere. Power consumption should be as minimum as possible.
Carbon emission	It is the carbon produced by all the system's resources. The LB schemes are highly applicable to reduce this metric by migrating the tasks from the underloaded VMs to other VMs and shutting down their whole system [65].
Transmission hop count	It specifies the network system's degree of congestion level and packet loss probability. In a higher hop count path, the intermediate devices are more likely to be congested, or the links are more likely to be the bottleneck. Thus, we have a higher packet loss probability and transmission delay. In a lower hop count path, the opposite happens. On the basis of this metric, along with overhead, end-to-end delay, PLR, and resources availability, an LB policy designer can design an optimized function for effectively balancing the system.
Flow completion time (FCT)	This metric describes the flow transmission efficacy in a system's long-lived traffic flows. It is an aggregate period taken by a flow to transfer a file completely.
Types of traffic	In data center systems, the traffic can be characterized as short-lived (i.e., web browsing or web queries) or long-lived (VMs migration and data transfer) flows. The short-lived flows have a shorter lifetime (duration), which must be transmitted before FCT. The long-lived flows (generated by applications) have a higher lifetime and usually require better throughput [66]. These two types of traffic flow run in a single network whose basic requirements are opposite. The short-lived flows require faster FCT, and the longer-lived flows require higher throughput. Hence, an LB mechanism must adapt its procedure when differentiating traffic types.

Table 1. *Cont.*

Metric	Description
Workload balance	Concerning SDN controllers, workload balancing is very much required. Here, the term 'workload' itself suggests the number of tasks to be completed by an SDN controller.
Peak load ratio	This metric can be assessed by estimating the traffic load of each link at a given time [62]. Any LB policy should consider this metric while performing workload and traffic scheduling in the system.
Re-association time	This metric [67] states the time taken to associate a client's device to the underloaded access point to ensure appropriate LB amongst access points.
Matched deadline flows	This metric defines the total proportion of flows sustaining its time delay deadline constraints. For instance, some flows running may need to arrive at the destination within a time constraint. Otherwise, the received information could be more beneficial [68].
Cumulative frequency	This metric suggests the precise traffic load information in the queue, which assists in maintaining an efficient LB in the system [68].
Percentage of all VMs to be positioned in host	This metric applies to VM LB mechanisms in data centers. It indicates the percentage distribution of VMs of multiple data centers as restraints. This metric's assessment has been done by utilizing the maximum and minimum percentage values of all VMs positioned in each cloud. That is how the allocation of these VMs is balanced amongst clouds [15].
Number of migrations	It is the number of tasks switching between multiple VMs. If a VM is overloaded, the tasks performed by it can be switched to other VMs with a lesser workload. If this switching is significant, the system's performance (i.e., migration and scheduling time overhead) will degrade significantly [15].

4. Related Surveys

4.1. LB in CC

An effective LB scheme solves many problems inherent in the structure and usage of the cloud. LB is aimed at two objectives: (1) task scheduling, and (2) resource provision in a distributed environment. The concept of effective task scheduling and resource provisioning leads to the following advantages:

1. Resource availability whenever required.
2. Efficient resource utilization during less and heavy load.
3. Controlled power consumption.
4. Reduction in resource utilization cost [69].

Due to extreme variations in nodes' available resources and capability, controlling the workload (and its efficient distribution amongst available nodes) is vital for maintaining the system's performance. A solution is an LB algorithm that forecasts the pattern of the arrival of users' tasks in the cloud. In such a dynamic distributed environment, dynamic LB algorithms perform far better than static algorithms. This occurs because the workload is dynamically transferred from a heavily loaded node to a lightly loaded one. As a result, the dynamic LB schemes assist in maintaining scalability and resource utilization in the cloud system [70]. Dynamic schemes are far more flexible and dynamically adaptive as these schemes consider various system attributes both preceding and during runtime. These schemes' operational behavior depends on the previously collected information about the working nodes in the cloud setup and their runtime attribute properties. Dynamic schemes need continuous monitoring of IoT nodes and their attribute properties. Thus, these schemes are much more complicated to implement [71]. In contrast, static LB schemes are steady while their implementation is relatively straightforward. This brings less complexity and overhead to the system. However, static LB policies fail to adapt and do not perform satisfactorily in a dynamic heterogeneous environment [70].

Katyal and Mishra [69] investigated LB mechanisms in static and dynamic cloud setups. They classified LB techniques concerning task dependencies and spatial node distribution, and then discussed their relative merits and demerits. Furthermore, they categorized spatial node distribution-based LB techniques into distributed, centralized, and hierarchical schemes. Randles et al. [72] presented some distributed LB schemes for the large-scale cloud environment. They discovered that (1) a nature-instigated scheme can be utilized for attaining a comprehensive LB through local server actions, and (2) self-regulation may be planned via arbitrary system sampling. The authors also revealed that there is a possibility to reorganize the system's structure to optimize task scheduling at the server end. Sreenivas et al. [73] surveyed sender-initiated, receiver-initiated, symmetric, static, and dynamic LB schemes. In a sender-initiated LB scheme, the sender initiates the process to find the under-loaded nodes when nodes get congested. In a receiver-initiated LB scheme, receiver or lightly loaded nodes look for heavily loaded nodes to share the workload. In symmetric LB, both sender-initiated process and receiver-initiated process techniques are combined. The authors discussed various LB metrics such as scalability, allied overhead, throughput, fault tolerance, response period, resource utilization, and performance. However, an analytical study could have been performed considering these metrics, which are missing in the abovementioned surveys. Furthermore, the authors discussed service provisioning, VM migration, server consolidation, power, and stored data management.

Raghava and Singh [74] reviewed LB techniques in the cloud. They discussed key parameters affecting LB, such as (1) geographical dissemination of nodes, (2) static versus dynamic behavior and complexity of schemes, (3) and traffic assessment over the different locations (geographical). Furthermore, they compared the schemes using various criteria such as scalability, complexity, resource utilization, response period, and fault tolerance. Milani and Navimipour [12] investigated hybrid and dynamic cloud-based LB techniques considering scalability, resource utilization, migration time, response time, throughput, and makespan metrics. This survey states that no unique LB scheme considers all the involved parameters (i.e., QoS, consistency, and scalability) during LB. Furthermore, the authors proved that the research in LB is ongoing toward the novel concept of decentralized LB. However, the authors did not consider task relocation and failure management in the design of an LB mechanism. Moreover, they could have discussed workload scheduling and LB schemes for IoT and MapReduce Hadoop more.

A load balancer allocates the user tasks to VMs according to their QoS requirements. When a physical machine gets overloaded, a few VMs can be shifted to a distant location (VMs migration). In the case of an overburdened node in the cloud, the additional workload on a system can be shifted onto some other available underburdened system (depending on that system's operational attributes). In the case of an underburdened node, its current workload can also be shifted to some other resource-full machine and can shift that underburdened node to an energy-saving mode. Such resource scheduling, rescheduling, or workload migration or shifting is realized through *VM migration* [75–77].

In the context of VM migration, Xu et al. [15] investigated surveys by examining the diverse characteristics such as scheduling situation, resource category, management methods, VM style distribution dynamicity, and consistency for LB in VMs. The authors presented various scheduling metrics to assess the performance of VM's LB techniques, such as standard deviation and load variance of utilization, overburdened hosts, makespan, throughput, connections, imbalance level/score, residual resources, and the number of migrations. They discussed various simulation tools and realistic platforms for testing VM's LB schemes. The authors also showed that metaheuristic approaches perform better than heuristic approaches. Nevertheless, so far, metaheuristic techniques have only been tested with simulation software. Hence, these approaches need to be tested in a realistic real-time environment, which ultimately suggests their possible future in a real cloud setup. An LB technique based on VM migration must optimize LB and curtail cost. There is a need to evaluate the tradeoff (if any) and maintain consistency across these multi-objective criteria. The authors also noted that knowing which technique is suitable for which environment is essential. Considering the variety of available surveys and the existing techniques, this problem is still open. There is also a serious need to test these techniques under a similar cloud setup. Here, the authors outlined that a relative performance assessment for these

VM's LB techniques under a matching and real cloud setup needs to be accomplished. The authors focused on various other issues, such as the organization of the scheduling model (i.e., future load prediction, analysis-based LB schemes, and communication overhead analysis) and VM models (i.e., self-regulatory VM LB techniques and integration of heuristic and metaheuristic methods).

Ghomi et al. [65] discussed LB and workload scheduling mechanisms. They categorized these proposals, i.e., Hadoop MapReduce, agent-based, natural phenomenon-based, application-oriented, network-conscious, workflow explicit, and general LB approaches. They addressed vital challenges in current LB technique design principles and outlined future research directions. The authors investigated their classified categories using various QoS-based metrics. Their survey gave deep insights into and an analysis of power management and Hadoop MapReduce schemes. Unfortunately, this survey lacks cluster-based and workload-based LB mechanisms that are needed to accomplish workload on limited available resources. Moreover, a better analysis is required concerning the communication overhead issue, which is also missing in this survey.

The surveys [78–80] considered cloud-based LB. Ahmad and Khan [79] investigated LB techniques and tools/platforms for their implementation. They also evaluated and shed light on their performances by considering metrics such as resource utilization, scalability, transfer time, fault tolerance, overhead, reaction period, and throughput. The authors shed light on the methodologies for fair load distribution amongst nodes for better resource utilization. Kumar and Kumar [78] addressed a state-of-the-art survey for vital LB schemes' concerns and challenges that must be taken care of while developing and implementing any LB algorithm. The authors discussed various challenges that researchers must consider while designing any LB policy, such as geographically distributed systems, single-point breakdown, VM migration, heterogeneous systems, scheme complexity (algorithmic), scalability of a load balancer, and storage management. They stated that the primary motivation of any scheme is to efficiently distribute load amongst all the available network links on systems clusters or devices to maximize their utilization of resources to improve the response period. Hence, it subsequently lessens the waiting time for a device. Lastly, the authors presented potential directions (i.e., appropriate resources distribution to workload, sustaining a proper balance between functional and nonfunctional requirements) in which much work is needed. Nevertheless, a deeper analysis of extra performance (i.e., communication overhead) and operational attributes is required, which is missing in these surveys [78,79].

4.2. LB in Software-Defined IoT

The conventional IP-based network architectures are highly rigid, and it is difficult for network engineers to make them adaptive to dynamic network conditions. The already installed conventional devices do not support potential improvements to these basic network architectures. Recently, the SDN networking model [39] was delved into. LB methods in software-defined IoT can be classified into two main categories including deterministic and non-deterministic approaches.

- A *deterministic* approach always produces the same output for specific input. Differential equations often describe its processes. Furthermore, the output of the model is completely specified by the values of the parameters and the primary situations. Deterministic approaches have used both distributed controllers and centralized controllers. Migration and rerouting are deterministic approaches that can reduce response time and overhead. They can also improve the system throughput and the degree of LB. However, these approaches may generate unacceptable energy consumption, unacceptable latency, unacceptable packet loss, and low availability. The determinist LB schemes do not evaluate many QoS parameters.
- *Non-deterministic* approaches: LB is an NP-complete problem and can be solved using a nondeterministic approach that can find a solution to the NP-complete problem in polynomial time. A nondeterministic LB approach may have different behaviors

on different runs for the same input. It is often applied to acquire approximate solutions when an exact solution is difficult (or costly) to acquire using a deterministic algorithm. These approaches use various methods such as greedy, metaheuristic, approximation, genetic algorithm, multi-objective particle swarm, and particle swarm. These methods can improve utilization and the degree of LB. Moreover, latency can be reduced. However, these approaches have high computational time and can generate unacceptable energy consumption, throughput, packet loss, and overhead. Notably, non-determinist LB schemes need to consider all the LB metrics.

Neghabi et al. [19] considered most SDNs' deterministic and nondeterministic LB schemes. Their study considered qualitative LB metrics such as synchronizations per minute, degree of LB, power consumption, throughput, delay, execution period, resource utilization, forwarding information, migration price, assured bit rate, overhead, peak load ratio, workload, and PLR. The authors showed that the existing techniques do not have a particular system to ascertain almost all the QoS parameters for LB judgments. Their survey stated that nearly all the initial efforts in LB had yet to discuss traffic shaping/patterns and data/control packet priorities. Additionally, planning a more poised LB-based QoS metric is still an open issue. This study revealed that the researchers should have investigated various problems such as power saving, cost of resource utilization, and carbon emissions. Hamdan et al. [66] and Belgaum et al. [81] suggested the latest LB schemes for the SDN environment. Neghabi et al. [19] reviewed SDN-based LB techniques published between 2013 and 2017. The authors improved their survey by depicting another perspective to readers by publishing another nature-inspired SDN-based LB survey [82] in 2019. However, their survey [82] considered only SDN-based LB techniques between 2013 and 2017. Hamdan et al. [66] and Belgaum et al. [81] suggested deep insights into SDN-based LB. Moreover, Belgaum et al. [81] indicated that a significant number of LB policies were published after the year 2017, and none of the surveys [19,82] discussed the techniques published after that. A few years ago, some survey papers [43,81,83,84] were published. Li et al. [83] and Zhang et al. [84] addressed several issues regarding LB in an SDN environment. They gave a new perspective to the reader on SDN-based LB for data centers. Nonetheless, these surveys did not consider the facet of LB in the case of the SDN networking environment. Hamdan et al. [66] suggested their deep insights compared to earlier surveys. Their study suggested an appropriate taxonomy for classifying available LB techniques in an SDN environment. This taxonomy covers multiple perspectives and aspects for the concerned area, such as the data plane, control plane, and objectives of LB mechanisms. Furthermore, the authors shed light on the performance and operational metrics utilized for LB techniques in the SDN environment. The authors discussed the challenges which need to be studied. These challenges are (1) managing heavier controller load in data plane schemes, (2) active LB practices in case of more than a single controller, (3) LB methods in case of hierarchical controller, (4) network virtualization inside controllers, (5) controller assignment methods, (6) flow regulation arrangement delay, and (7) network management.

4.3. LB in the IoT Environment

IoT permits the orchestration and coordination of physical web-enabled intelligent objects. These smart objects (e.g., sensors) may be in an isolated locale where human intervention is not possible; there may be a chance that these devices may be part of an unstable network environment. Moreover, they may work on their limited capability due to power constraints issues. One of those devices can also act as an IoT gateway. An excessive load on the links (network) directly/indirectly associated with such a gateway or extreme load-induced failure at the gateway itself ultimately results in temporary or even permanent unavailability of the IoT service [85]. Since these gateways act as an intermediary between exceptional sensing infrastructure-based networks and data centers, the LB policy should first ensure that such gateways are not overloaded. Otherwise, the network could get detached or partitioned. An extensive survey is needed to recognize the available LB procedures. These procedures should be assessed through performance and operational metrics.

IoT objects have restricted computing power and storage. Many researchers considered these restrictions while designing an LB strategy in *low-power and lossy networks* (LPLNs) [86–89]. The researchers concentrated on enhancing scalability and lessening the power consumption in LPLNs. The preliminary routing protocol for LPLNs (called rp-LPLN) is commonly accepted. The research community accepted the rp-LPLN protocol as a revolution, while some researchers argued that it had many major flaws. Many weaknesses were revealed, such as propensity concerning load imbalance, instability, and a lesser emphasis on mobility, which needs to be emphasized.

Load imbalance is the major drawback of rp-LPLN [90]; the rp-LPLN protocol deals with uneven traffic distribution in dense LPLN environments. Generally, rp-LPLN adopts a proactive distance vector routing mechanism in which nodes are positioned and build a destination-oriented directed acyclic graph (DoDAG). rp-LPLN builds a routing topology (i.e., DoDAG) rooted at a single or multiple LPLN border router (LPLN-BR) [89,91]. rp-LPLN also provides IPv6-based communication (bidirectional) between devices. Moreover, the rank associated with a node indicates its concrete position from the root of DoDAG and in the LPLN. During DoDAG construction, the newly arrived nodes (i.e., those not yet part of the graph or LPLN) choose the parent on the basis of their ranking parameter. However, there is a dilemma; all the nodes (leaves in DoDAG) may choose the same parent and stay away from other available parent nodes. This problem in such a network environment is termed '*the thundering herd*' [92]. Hence, the idea of DoDAG construction via parent selection notion in the rp-LPLNs leads to the issue of imbalance. Furthermore, these parent nodes are also low-powered constrained devices, which may assist in the network as a forwarder. Therefore, there is a high probability of a parent node's resources depleting more rapidly than leaf nodes. These issues are characterized as *hotspots* and *bottlenecks*. The authors [81] focused on rp-LPLNs-based LB mechanisms and revealed vital topics such as hotspot, thundering herd, bottleneck, congestion-induced unbalanced load on nodes (leaves and roots), instability, and poor delivery ratio.

Ghaleb et al. [88] analyzed LPLNs/rp-LPLNs-based schemes. They discussed necessary details regarding the environment, the technology utilized for communication, and particular forwarding requirements in LPLNs. They also pointed out several shortcomings in such procedures. The authors discussed limitations regarding rp-LPLN, such as route selection/maintenance procedure and downward forwarding. This survey gave readers in-depth insight into numerous studies suggested to improve rp-LPLNs performance. It revealed an exhaustive comprehension of multiple forwarding necessities based on rp-LPLNs. It shed light on issues such as downward traffic patterns, single- versus multiple-instance optimization, metric composition, LB, real-time testbed implementation, and deployment that can be worked upon in the future.

The surveys [88,91,93] discussed LB issues. Nevertheless, their focus (especially [88,93]) was partially on LB. Furthermore, there are other surveys [94–97] whose focus was not on LB at all. Some of these surveys discussed other issues, such as mobility extensions [94], resource management [95], and security [97] in the context of rp-LPLN environment. Kim et al. [91] analyzed numerous research attempts focused on rp-LPLNs. They suggested a topic-oriented survey for multiple research attempts to improve rp-LPLNs performance. Moreover, they emphasized the issue of rapid power depletion when the load distribution on the network becomes imbalanced. They highlighted that the rp-LPLN protocol must be energy-efficient as the nodes in the LPLNs operate on very-low-powered batteries. Maximizing a node's lifetime in such a scenario is essential. Hence, rp-LPLNs must stabilize the amount of traffic load that needs to transfer on each node to provide proper power management amongst nodes. Kim et al. [91] further stated that in the case of across-the-board applications such as smart grids and home automation, there is a high chance that devices/nodes nearest to LPLN-BR have to transfer huge amounts of data even if other devices/nodes generate a lesser amount of data. The ultimate objective of rp-LPLN design standard is to provide the interconnection of many devices, which generate massive network traffic. Hence, there is a dire need to assess the LB procedures under severe congested situations. Initially, RFC 6550 [89] favors the exploitation of reactive approaches (rather than proactive approaches), justified the context of power efficiency behind favoring this procedure. Kim et al. [91,98] shed light on the design and implementation of a standard followed in RFC 6550. LB is dissociated from this original standard. Later, two more new companion standards (namely, OF0 [99] and MRHOF [100]) were suggested, which were aimed at handling this issue. Nonetheless, various studies stated that these rp-LPLN-based companion standards tend to create a load imbalance in the network. Precisely, the conventional rp-LPLN with companion standards (mentioned earlier), along with the expected transmission count (ETX) parameter for parent node selection and the hop count parameter (for ranking), generates load imbalance since such metrics are only focused on identifying a parent node with adequate link quality. Subsequently, this leads to unbalanced power depletion among nodes and the creation of bottlenecks at some portion of the network.

Kharrufa et al. [101] emphasized the prominence of rp-LPLN and suggested a systematic review in the context of IoT. They presented deep insights into the proposed procedures that improve rp-LPLN performance. They assessed each approach's efficiency concerning applicability, scalability, throughput, flexibility, power efficiency, and end-to-end latency. The reviewed policies were categorized according to the field of improvement, such as QoS, mobility, congestion control, security, and power consumption. Pancaroglu and Sen [87] surveyed rp-LPLN-based LB mechanisms by categorizing them into two clusters:

1. Mechanisms utilizing objective function and routing parameters to improve the load balance procedure.
2. Studies (in the concerned area) based on heuristic schemes.

The authors categorized the first cluster into two subclusters: attempts based on conventional parameters and attempts based on customized parameters. They assessed the considered attempts by shedding light on critical parameters such as ETX, remaining power, hop count, and novel routing parameters as suggested in considered heuristic procedures. The authors explained some shortcomings of the related studies. Such shortcomings are the limited assessment in real-time testbeds/platforms. In most cases, the considered network size was small (limited number of devices/nodes), and many studies need to elaborate much in the context of network reliability through evaluation results. They also lacked standardized processes while selecting performance metrics, selected inferior quality paths, and presented smaller or no exploitation of multiple examples.

4.4. LB in Heterogeneous (IoT/Fog/Edge) Environment

Data generated by IoT devices can be directly transferred to the data centers using cloud-based services. However, as the size of the data increases, so will the complexity of sending those data from the IoT device to the cloud. Such transfer may become infeasible in a limited bandwidth network environment or due to security apprehensions. Meanwhile, location-aware and delay-sensitive IoT applications must receive location-related data well on time. A remote cloud setup cannot satisfy such delay constraints. A specialized computing paradigm is needed that can run more closely to the connected devices, which can address issues such as limited bandwidth in a network environment, geographically detached, delay-sensitive, and security subtle applications.

Fog computing (FC) encompasses the CC paradigm to the system's edge. FC conveys computing and networking supremacy into the system's edge, nearer to intelligent IoT devices/nodes and end-users, because of being sustained by prevalent fog nodes. FC extends the computing procedure to the system's edge instead of accomplishing the IoT services in the cloud platform [102–104]. FC is a virtualized platform that bridges the gap between IoT devices and the cloud by offering storage, computational services, decision making, data management, and network communication facilities between conventional CC-based data centers and end-users. Hence, as data move from IoT devices to the cloud, these features happen along the path between the cloud and the IoT devices. In an FC environment, the routers can act as specialized servers. These routers can augment performance in terms of storage and computing, which can act as computing nodes [104,105]. Nonetheless, Al-khafajiy et al. [106] showed that the notion of fog in the FC paradigm is also insufficient to manage these smart IoT devices' temporal and spatial dissemination. This leads to an unbalanced LB amongst fog nodes. Furthermore, VMs become overburdened on the fog layer with a steady increase in user requests. Thus, an LB mechanism is required to handle such a scenario on the fog layer.

Dynamic resource scheduling and resource allocation policies must be introduced while managing data centers. Such efficient policies assist in improving the overall resource utilization and attaining appropriate LB in the data centers. In the case of the CC paradigm, the ultimate objective of any resource allocation mechanism is to maximize the figure of active devices. The policy of resource allocation should be so effective that the maximum number of devices can work on limited resources. Another aim of such an approach is to maintain workload stabilization among all the actively working devices to evade overloaded/underloaded resource usage and bottleneck formation.

Memon et al. [107] suggested a multilevel dynamic traffic LB protocol (MDTLB) for data centers. MDTLB addresses the problem of constantly varying network parameters (i.e., continuously varying bandwidth, rapidly changing delay, middlebox failures, and topological changes) that data centers face. The authors proposed the adaption of imperative networking parameters while performing LB. Typically, MDTLB utilizes roundtrip time (RTT) to define the network link's state clearly. However, Carlucci et al. [108] proved that, when the RTT estimations are utilized as a network congestion metric, a lesser channel utilization may be attained in the existence of reverse traffic. Xu et al. [105] showed issues related to the design challenges of resource scheduling and resource allocation mechanisms in an FC environment. They shed light on the operational-level complexity of these mechanisms in the FC environment since there may be a chance that the computing nodes could answer an application both in the case of clouds and fog. The computing nodes are well distributed on the network's edge when we consider the perspective of the FC environment. However, these nodes are distributed in a unified data center in the cloud. Xu et al. [105] stated that the IoT services/application resource demands could diverge for the computing nodes. These services may have different storage capability requirements, bandwidth, and computing power requirements. Therefore, there is a dire requirement to undergo resource allocation and resource scheduling mechanisms for dynamically resource-demanding IoT services and applications to accomplish LB.

Vaquero and Rodero-Merino [109] offered an explanation and formal definition of fog and related technologies in the context of the FC paradigm. Their attempt covered the challenges allied with the FC paradigm that need to be addressed so that investigators, implementers, and designers can disclose its full potential. Yi et al. [104] deliberated further on the formal definition of FC, presented demonstrative application situations, and shed light on challenges and concerns concerning the design and implementation of the FC system. Chiang et al. [110] discussed the differences between CC, FC, and EC, the exclusive features offered by fog, and fog's role in the context of security and privacy. However, the authors discussed these issues very abstractly. Chiang et al. [111] initially presented emerging challenges in IoT. They explained how difficult it is for the current networking and computing paradigms to address those emerging challenges. They discussed why there is a need for new networking and computing paradigms such as FC. However, there is a critical requirement for the performance analysis of reviewed technologies, algorithms, architectures, models, and schemes from the perspective of definite and well-motivated parameters, which is missing in the above surveys altogether. Mouradian et al. [53] stated that these surveys did not ponder algorithmic perspectives much, although they play a serious role in the FC environment. Furthermore, they surveyed numerous architectural and algorithmic-based works. They discussed the algorithmic and architectural point-of-view in the fog system design. They also highlighted challenges and concerns concerning fog-based architectures and algorithms. Nevertheless, a deeper analysis of some more performance (i.e., communication overhead) and operational attributes is required, which is missing in these surveys [105,109–112].

Aazam et al. [112] reviewed the task offloading mechanisms in CC, IoT, and FC environments. The idea of offloading can be implemented via various aspects such as LB, computational needs of an application, delay, and power management. The authors presented a taxonomy concerning such techniques and highlighted the types (e.g., cloudlet and mobile EC) and middleware technologies' requirements for task offloading. They deeply analyzed numerous criteria such as power consumption, LB, and delay utilized in offloading. However, a separate study on LB is missing in this survey. Fricker et al. [113] utilized the idea of LB to activate task offloading within fog data centers. The authors considered a basic scenario where data centers are installed at the network's edge. The authors presumed the case of an overburdened data center receiving a request, which is then forwarded to a comparatively lesser loaded neighboring data center with some possibility. That is how a lesser-loaded adjacent data center can handle a rejected request from an overloaded data center. They assessed the achieved gain when neighboring data centers collaborate in a fog context. Mukherjee et al. [114] highlighted that the terminal devices/nodes in an FC environment might dynamically perform transitions from one state to another (e.g., active or inactive). These devices/nodes often join or leave any FC instance dynamically. Due to vigorous requirements of resource distribution and high communication overhead, achieving LB in such an environment is challenging. Moreover, the authors initially gave in-depth insights into FC architecture. They abridged various resource and service allocation mechanisms intending to address critical concerns such as bandwidth, delay, and power consumption in the FC context. They also highlighted challenges and problems such as application offloading, resource management, heterogeneity, fog radio-access networks, SDN-based FC paradigm, and standardization. Ghobaei-Arani et al. [115] reviewed FC-based resource management policies. They presented a standard taxonomy using metrics such as workload offloading, application placement, LB, resource allocation, scheduling, and managing. They showed a comparison analysis of the resource management policies using specialized schemes, performance metrics, case studies, and evaluation tools involved.

Hong and Varghese [116] addressed FC-based resource management requirements and their implementation challenges. The resources at the network's edge are constrained in terms of computational resources since edge-based intelligent devices have lesser processing capability due to smaller-size processors and limited power. Moreover, the processors' architecture difference amongst edge devices, the high application-level contentions for limited resources, and dynamic workload variations make resource management a complex task that policy designers consider while dealing with FC and EC scenarios. Consequently, the authors gave deep insights into architectures, algorithms, and schemes based on FC and EC. The authors also highlighted the types of techniques (i.e., cooperative, graph-based, optimization schemes, and breadth-first search) utilized by LB mechanisms at the edge of the network.

Table 2 presents important surveys that summarize LB research in IoT.

Figure 7 presents LB performance metrics in the discussed computing and networking paradigms.

Table 2. A comparison of surveys that summarize LB research in IoT.

Year/Ref.	Techniques Presented	LB Metrics Discussed [1]	Suggested Future Directions	Tool(s)/Testbed(s) Discussed	Weakness	Coverage
Surveys on CC-based LB						
2019 [78]	Static and dynamic LB techniques.	(i), (iii), (iv), (v), (vi), (vii), (viii), and (xiii)	Power saving, resource utilization cost, and carbon emissions.	Cloud Simulators (simulation tool)	An analysis of some extra performance (i.e., communication overhead) and operational attributes is missing.	2010–2018
2018 [79]	Fair load distribution for better resource utilization methodologies.	(i), (ii), (iii), (iv), (v), (vi), and (x)		CloudAnalyst, GroudSim, GreenCloud		2008–2016
2017 [15]	Centralized and distributed load schedulers for VMs placements.	(iii), (ix), (xiii), (xxi), (xxii), (xxiii), (xxiv), (xxv), and (xxvi)	Realistic implementation of metaheuristic approaches.	Realistic Platforms: ElasticHosts, OpenNebula, Amazon EC2 (web service) Simulation toolkits: FlexCloud, CloudSched, CloudSim.	These attempts do not discuss much regarding workload scheduling and LB schemes for IoT.	2008–2016
2017 [77]	Statistics-based and nature-inspired LB techniques.	(i), (iii), (v), (vi), (ix), and (xii)	Refinement in algorithms to ascertain migration cost, active thresholding, and communication overhead to accomplish LB efficiently. Efficient workload prediction schemes. Current schemes do not address and focus on the effective usage of available limited network resources. Algorithms should be tested over real-time testbeds and a cloud scenario.	×	The issue of high communication overhead needs to be discussed in detail, which is missing from this survey.	2007–2017
2017 [65]	Task scheduling and LB (schedulers in Hadoop, MapReduce optimization, agent-based, natural phenomenon-based, and general schemes).	(i), (iii), (iv), (v), (vii), (viii), (ix), (xii), (xiii), and (xiv)	A tradeoff between various LB metrics. Proposing an LB scheme that could improve as many parameters/metrics as possible. Resource utilization for processing from multiple clouds (providers). Power consumption and carbon emissions.	×	This attempt lacks cluster-based and workload-based LB mechanisms, for accomplishing workload on limited available resources.	2008–2017
2016 [12]	Hybrid and dynamic cloud-based LB.	(i), (iii), (v), (vi), (viii), and (ix)	Decentralized LB, task relocation, and failure management features.	×	It superficially discusses workload scheduling and LB schemes for IoT.	2010–2015
2014 [73]	Sender-initiated, receiver-initiated, symmetric, static, and dynamic LB techniques.	(i), (ii), (iii), (iv), (v), (vi), and (vii)	Power and stored data management, server association, programmed service provisioning, VMs migration, and carbon emission rate.	×	This survey does not identify or consider the shortcomings of the assessed LB techniques.	2010–2012
2014 [69]	Task dependencies and spatial node distribution (distributed, centralized, and hierarchical schemes) in LB schemes.	×	×	CloudSim [68]	An analysis of QoS performance metrics and operational attributes is missing.	2007–2012

Table 2. *Cont.*

Year/Ref.	Techniques Presented	LB Metrics Discussed [1]	Suggested Future Directions	Tool(s)/Testbed(s) Discussed	Weakness	Coverage
2014 [74]	Issues (i.e., spatial nodes' geographical distribution and their traffic pattern analysis, dynamic/static behavior, and complexity of algorithms) affecting LB mechanisms.	(i), (ii), (iii), (iv), (v), (vi), and (viii)	×	×	An analysis of some more performance (i.e., communication overhead) and operational attributes is missing.	2000–2011
2012 [71]	Static and dynamic LB mechanisms	×	×	×	An analysis of QoS performance metrics and operational attributes is missing.	2008–2012
Surveys on SDN-based LB						
2021 [66]	Control plane and data plane-based LB methodologies.	(ii), (iii), (v), (vi), (xv), (xvi), (xvii), (xviii), (xix), and (xx)	Managing heavier controller load in data plane schemes. Active LB practices in case of more than a single and hierarchical controller. Network virtualization inside controllers, controller assignment methods, flow regulation arrangement delay, and network management.	×	The specific comments regarding tools, testbeds, and simulation platforms are missing.	2008–2020
2020 [81]	Conventional and artificial intelligence-based schemes.	(ii), (iii), (v), (vi), (viii), (xii), (xiii) (xv), (xvi), (xxii), (xxv), (xxviii), (xxix), and (xxx)	Traffic-aware LB mechanisms, reduction in communication latency when the center controller becomes congested, power-efficient LB schemes, and network function virtualization assistance to cloud users.	×		2015–2019
2020 [43]	LB in controller, server, and wireless links. Communication path selection and AI-based LB.	(iii), (xvi), (xxxvii), (xxxviii), (xxxix), (xl),and (xli)	Energy preservation issue while designing LB scheme for SDN. Smart LB schemes (in SDN) require node and link failure consideration. Adaptations of such schemes in 5G environment.	Mininet, OMNET++, IPerf, MATLAB, Python, .Net, Maxinet (tools and emulators)	This survey lacks in discussing fast adaptive rerouting and multipath solutions, which are also important areas of concern, especially when the network's links fail rapidly. Moreover, in [48], the authors do not discuss much regarding tools, testbeds, and simulation platforms.	2007–2020
2019 [82]	Nature-stimulated metaheuristic techniques.	(ii), (iii), (v), (vi) (xiii), (xv), (xvi), and (xxii)	Traffic shaping, its pattern, and packet priorities can be utilized in the future. Power consumption and carbon emission should be addressed in future nature-inspired metaheuristic load-balancing proposals.	×		2013–2017
2018 [19]	Deterministic and nondeterministic LB techniques in SDN.	(ii), (iii), (vi), (viii), (xii), (xiii), (xvi), (xv), (xxi), (xxii), (xxiii), (xix), (xxiv), (xxv), (xxvi), and (xxvii)	Planning a more poised LB-based QoS metric, power saving, resource utilization cost, and carbon emissions.	×	The issue of communication overhead needs to be discussed in the context of SDN in detail, which is missing from this survey.	2013–2017
2017 [83]	Centralized and distributed SDN-based LB schemes.	×	×	×	This survey does not consider the shortcomings of the assessed LB techniques. It does not present specific comments regarding tools, testbeds, and simulation platforms. No performance and operational metrics-based comparison is made in this survey.	2009–2017

Table 2. *Cont.*

Year/Ref.	Techniques Presented	LB Metrics Discussed [1]	Suggested Future Directions	Tool(s)/Testbed(s) Discussed	Weakness	Coverage
Surveys on LB in LPLNs-based IoT						
2021 [87]	LB mechanisms for rp-LPLNs-based IoT.	(iii), (xii), (xv), (xvi), (xvii), (xlii), (xliii), and (xliv)	Concerning LB, the following are the challenges and open issues allied with rp-LPLNs: power consumption, stability, reliability, mobility, objective function, and congestion.	×	This survey lacks in discussing fast adaptive rerouting and multipath solutions, which are also important areas of concern, especially when the network's links fail rapidly.	2009–2020
2019 [101]	Significant attributes of rp-LPLN and various pros and cons of rp-LPLN in numerous IoT applications are discussed. rp-LPLN improvement for power efficiency, mobility management, QoS, and congestion control.	×	There is not a variant of rp-LPLN which succeeds optimality in all IoT applications. Thus, in the future, rp-LPLN's functionality must be improved so that it will work and provide optimum performance for most IoT applications. The original rp-LPLN standard ignored mobility management and congestion control completely. Thus, researchers must consider them while designing improved algorithms for rp-LPLN.	×	This survey systematically discusses the rp-LPLN-based routing protocol's performance-enhanced mechanisms. However, a detailed discussion on LB mechanisms is missing.	2010–2017
2019 [117]	LB and congestion control schemes in the wireless sensor networks context. Schemes were classified on the basis of several criteria, e.g., routing metrics, cross-layer design, and path variety.	×	To design a more efficient scheme considering how to reduce resource overutilization. Real-time/testbed implementations and deployment. Contemplation of numerous rp-LPLN. More solid experimental conclusions are required when integrating TCP with rp-LPLN. Cross-layering-based schemes must be proposed.	×	Since this attempt addresses congestion as an essential factor concerning deprived rp-LPLN's performance, there is a need to specifically evaluate the communication overhead in terms of normalized routing and MAC load, which is missing from the survey.	2009–2019
2020 [118]	Centralized and distributed load scheduling.	(i), (ii), (iv), (xii), (xv),(xvi), (xlix), (l), and (li)	Considering traffic shaping/patterns and data/control packet priorities, QoS parameter investigation for LB, and power saving, resource utilization cost, and carbon emissions is required.	×	The specific comments regarding tools, testbeds, and simulation platforms are missing. The issue of communication overhead needs to be discussed in the context of IoT in detail, which is missing from this survey. This survey does not discuss rp-LPLN's performance issues.	2009–2019
2018 [88]	Routing optimization, routing maintenance procedures, and downward routing.	(xii), (xvii), (xliii), (xliv), (xlv), and (xlvi)	Downward traffic forms, LB, single vs. multi-instance optimization, metric composition, real-time assessment via testbeds, and RPL deployment.	×	Although this attempt provides deep insight into the shortcomings of rp-LPLN, the focus is still not on LB schemes. Instead, the authors concentrated on routing procedures' (path selection and maintenance) optimization, which indirectly affects LB.	2011–2017
2018 [90]	LB problems in rp-LPLN such as hotspot, bottleneck, thundering-herd, instability, increased load on nodes, and low PDR.	(xlvii), and (xlviii)	Designing a more composed LB metric is still an open issue in RPL-based IoT-based networks.	×	It does not consider the shortcomings of the LB techniques. Specific analysis regarding tools, testbeds, and simulation platforms is missing. No performance metrics-based comparison is made.	2012–2018

Table 2. *Cont.*

Year/Ref.	Techniques Presented	LB Metrics Discussed [1]	Suggested Future Directions	Tool(s)/Testbed(s) Discussed	Weakness	Coverage
Surveys on LB in heterogeneous (IoT/Fog/Edge) environment						
2022 [119]	LB in advanced heterogeneous networks (HetNets) and machine learning (ML)-based LB methodologies.	(iii), (xxxviii), (lii), (liii), and (liv)	Incorporation of next-level ML procedures for planning LB schemes. Deep and transfer learning-based LB schemes. NFV/SDN, unmanned aerial vehicle (UAV) base station's (BS) dynamic deployment to improve LB performance in HetNets.	×	The specific comments regarding tools, testbeds, and simulation platforms are missing. This survey does not consider optimization and computational complexity for ML approaches.	2013–2021
2022 [102]	LB in fog computing environment. hybrid, precise, fundamental, and approximate LB methodologies.	(i), (iii), (v), (vi), (xii),(xxvii), (li), (lv), (lvi) and (lvii)	Systematic study of the problems such as power saving, multi-objective optimization, context-aware computing, green Fog, NFV/SDN, social networks analytics, and interoperability.	Mininet, CloudSim, MATLAB, iFogSim, CustomSimulator, Java platform, Work-robots, NS-2/3, Jmeter, CloudAnalyst, CPLEX/AMPL, and Scyther	Although the authors give a percentage of tools utilized so far in the literature, their relative analysis is missing in the survey. The issue of communication overhead needs to be discussed in the context of fog in detail, which is missing from this survey.	2013–2021
2021 [120]	LB and resource management in the fog computing environment.	(v), (vi), (vii) (xii), (xvi), (lv), and (lvi)	Load scheduling in fog. Testing of fog-based load balancers in a real-time environment. Power-aware resource utilization-based load balancers' design.	×		2013–2020
2020 [121]	The performance demands of ad hoc IoT networks. The authors contemplated the impetus for clustering as follows: LB, reducing power depletion, and refining connectivity.	(i), (iii), (iv), (xii), (xi), (xv), (xvi), (xlviii),(li),(lviii), and (lix)	Big challenges ahead when assimilating clustering with edge and fog such as resource provision improvement, and computational offloading. Challenges allied when combining clustering with 5G. Heterogeneity, interference, dynamicity, and scalability. Hierarchical management.	×	Since clustering was the main point of consideration in the context of WSNs, this attempt does not talk much about rp-LPLNs-based proposals, which is essential while dealing with LB in an IoT system. However, those internal details are missing from this survey.	2000–2019
2020 [113]	Resource management in the fog context: application placement, resource scheduling, task offloading, LB, and resource allocation.	(i), (iii), (vi), (xii), (xvi), and (lvi)	Resource management in the FC paradigm (open issues): power consumption, interoperability, scheduling and offloading, mobility, and scalability.	×	Although this survey gives a percentage of tools utilized so far in the literature, its relative analysis is missing. The issue of communication overhead is not discussed in the context of fog.	2014–2019

Table 2. *Cont.*

Year/Ref.	Techniques Presented	LB Metrics Discussed [1]	Suggested Future Directions	Tool(s)/Testbed(s) Discussed	Weakness	Coverage
This survey	It covers several LB issues from different perspectives: (1) dynamic LB and resources management in the cloud, edge/fog computing environments, (2) LB solutions for multipath communication in IoT, and (3) LB for distributing incoming traffic across a cluster of brokers. This survey has an advantage over the previous surveys.	It discuses many LB parameters	Traffic-aware LB, data prioritization, multi-objective optimization in LB decisions, optimal LB solutions, LB based on context-aware computing, interoperability, and efficient load management in vehicular fog computing	×	This comprehensive survey does not identify and review all the existing LB schemes for clusters of MQTT brokers.	2016–2022

[1] **Explanations:** (i) scalability, (ii) overhead, (iii) throughput, (iv) fault tolerance, (v) response time, (vi) resource utilization/resource utilization ratio, (vii) performance, (viii) migration time/price, (ix) makespan/capacity makespan, (x) transfer time, (xi) multi-sink support, (xii) power management/consumption, (xiii) degree of imbalance/LB, (xiv) carbon emission, (xv) packet loss/rate/delivery ratio, (xvi) end-to-end delay/delay/latency, (xvii) transmission hop count, (xviii) flow completion, (xix) servers root-mean-squared error, (xx) type of traffic, (xxi) synchronization/min, (xxii) execution time, (xxiii) forwarding information, (xxiv) guaranteed bit rate, (xxv) peak load ratio, (xxvi) the workload was performed by the controller, (xxvii) matched deadline flows/deadline, (xxviii) downlink/uplink rate, (xxix) reassociation period, (xxx) concurrency, (xxxi) standard deviation (stdev) and load variance of utilization, (xxxii) number of overburdened hosts, (xxxiii) percentage of all VMs to be positioned in host, (xxxiv) stdev of connections, (xxxv) stdev of outstanding resource, (xxxvi) number of migrations, (xxxvii) fairness, (xxxviii) QoS, (xxxix) congestion control, (xl) complexity, (xli) interference mitigation, (xlii) link quality level, (xliii) expected transmissions, (xliv) link color, (xlv) packet forwarding indication, (xlvi) stability index, (xlvii) network efficacy and stability, (xlviii) LB parameters, (xlix) heterogeneity, (l) network lifetime, (li) reliability, (lii) signal-to-interference noise ratio, (liii) physical resource block utilization, (liv) call drop/call block/outage, (lv) processing time, (lvi) cost, (lvii) availability, (lviii) physical layer support, and (lix) coverage, connectivity, and mobility management.

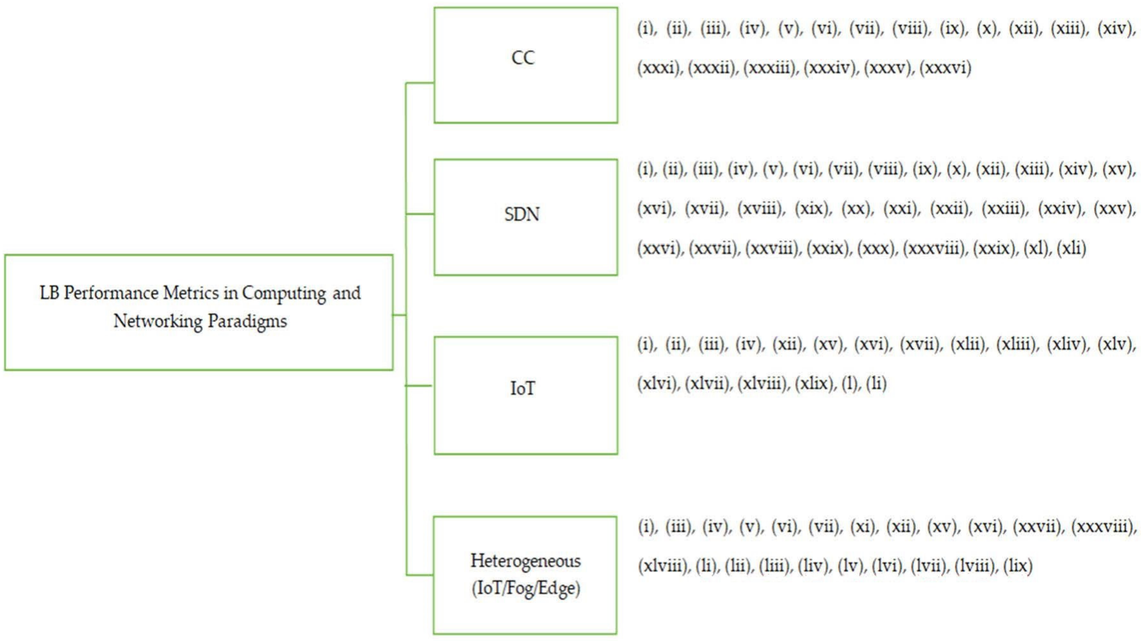
LB Performance Metrics in Computing and Networking Paradigms
CC
(i), (ii), (iii), (iv), (v), (vi), (vii), (viii), (ix), (x), (xii), (xiii), (xiv), (xxxi), (xxxii), (xxxiii), (xxxiv), (xxxv), (xxxvi)
SDN
(i), (ii), (iii), (iv), (v), (vi), (vii), (viii), (ix), (x), (xii), (xiii), (xiv), (xv), (xvi), (xvii), (xviii), (xix), (xx), (xxi), (xxii), (xxiii), (xxiv), (xxv), (xxvi), (xxvii), (xxviii), (xxix), (xxx), (xxxviii), (xxix), (xl), (xli)
IoT
(i), (ii), (iii), (iv), (xii), (xv), (xvi), (xvii), (xlii), (xliii), (xliv), (xlv), (xlvi), (xlvii), (xlviii), (xlix), (l), (li)
Heterogeneous (IoT/Fog/Edge)
(i), (iii), (iv), (v), (vi), (vii), (xi), (xii), (xv), (xvi), (xxvii), (xxxviii), (xlviii), (li), (lii), (liii), (liv), (lv), (lvi), (lvii), (lviii), (lix)

Figure 7. LB performance metrics in the discussed computing and networking paradigms.

5. Dynamic LB Techniques in IoT

A dynamic LB algorithm continuously monitors a node's load and estimates its workload by exchanging load and status information between nodes at regular time intervals. Then, the workload is redistributed across the nodes. In a distributed LB algorithm, all nodes participate in LB and maintain an information base (for sharing) to distribute and redistribute tasks efficiently. Furthermore, distributed algorithms can be cooperative or non-cooperative. A distributed cooperative LB algorithm allows all nodes to work together to achieve a common LB decision. In a non-distributed LB algorithm, a single node often decides on load distribution. Moreover, the behavior of non-distributed algorithms can be either centralized or semi-distributed. In centralized approaches, LB and all load distribution tasks are carried out by a single node. With centralized algorithms, fault tolerance is a problem. In the event of a single-node failure, node information may be lost. In the semi-distributed LB approaches, node clusters are formed, and each cluster functions as a centralized technique.

5.1. Classification of Load-Balancing Techniques

From the above discussion, the classification in Figure 8 emerges.

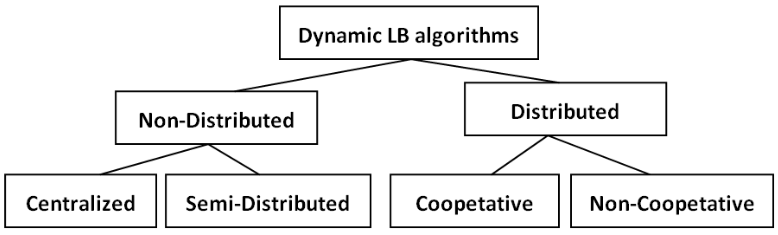

Figure 8. Classification of dynamic LB algorithms.

Table 3 also classifies dynamic LB methods depending on the computing paradigm adopted in IoT. As shown in Table 3, dynamic LB techniques for CC are classified into (1) general LB, (2) natural phenomenon-based LB, (3) agent-based LB, (4) task-based LB, and (5) cluster-based LB. Dynamic LB schemes for the fog are classified into (1) approximation algorithms, (2) exact algorithms, (3) fundamental algorithms, and (4) hybrid algorithms. This table is a comprehensive reference for researchers toward further and future work.

Table 3. Classification of dynamic LB techniques.

Classification Descriptor
Dynamic LB techniques for CC
General LB
Natural phenomenon-based LB
Agent-based LB
Task-based LB
Cluster-based LB
Dynamic LB in the Fog
Approximation algorithms
Exact algorithms
Fundamental algorithms
Hybrid algorithms

Some LB algorithms are agent-based LB [122], honeybee behavior-inspired LB [123,124], ant colony optimization [125], and throttled [126].

5.2. Load-Balancing Policies for CC, EC, and FC

Dynamic LB algorithms adopt four basic policies for executing user tasks [127]:

- *Task Selection Policy*: This policy identifies and selects the tasks that should be moved from one node to another. Such selection is based on the time needed to complete the task, the number of nonlocal system calls, and the amount of overhead required for migration.
- *Location Policy*: With this policy, tasks are transferred to underutilized (or free) computing nodes so they can process them. This policy selects the destination node via one of the three accessible methods (randomness, negotiation, and probing) and assesses the availability of necessary services for task transfer. The location policy selects the destination at random and transfers the task in the random approach. The destination is chosen by a node querying other system nodes in the probing strategy. The negotiation strategy involves nodes negotiating LB with one another.
- The *Transfer Policy*: This policy describes the conditions under which tasks must be transferred from one local node to another local or remote node. To determine the tasks that need to be transferred, two approaches, '*all current tasks*' and '*latest received task*', are used. In the 'last received task' approach, the task that arrives last is transferred once all incoming tasks have entered the transfer policy. Depending on the workload of each node, a transfer policy, based on a rule, determines whether a task has to migrate (task migration) or be processed locally (task rescheduling) in all existing task methods.
- *Information Policy*: This dynamic LB policy maintains all resource information in the system so that other policies can use it when making choices. It selects when to gather information. Agent, broadcasting, and centralized polling are three different techniques for gathering data from the nodes. In the broadcasting method, every node broadcasts its data, making it available to other nodes. Nodes now gather information using the agent approach. The demand-driven policy, periodic policy, and state change-driven policy are some examples of information policies.

All of these dynamic LB policies relate to the first processing of tasks by the transfer policy as they enter a system. After processing, a policy determines whether or not the tasks need to be moved to a remote node. The location policy determines the destination node that is idle or underloaded for the tasks that need to be transferred. The task is added to a queue and executed locally if a distant node is not available for execution. The information needed to make the selection is gathered via the location and transfer policies from the information policy.

5.3. The Function-as-a-Service(FaaS) Paradigm

A cloud function is a microservice that can only execute one action in response to an event. Such a function is a compact, stateless, on-demand service with just one functional duty. Depending on the application's objectives, the function implements particular business logic. There are three key aspects of cloud functions [128]:

1. Cloud functions are short-lived; each function usually only requires tiny inputs and delivers outputs after a brief period of time, making them simple to automate (for example, they can be easily auto-scaled).
2. A cloud function lacks operational logic since all operational concerns are transferred to the platform layer (operational and cloud-managed), enabling platform independence for the cloud function.
3. A cloud function has no regard for the context in which it is employed.

The function-as-a-service (FaaS) paradigm [129] is a cloud computing model that makes it possible for developers to construct, monitor, and call cloud functions even if they have little to no operational logic experience. Employing an FaaS model at the edge, the application can be decomposed into functions invoked individually or in a chain. FaaS is suitable for many IoT applications of practical interest with respect to the programming model (functional event-based), an efficient utilization of resources (both at the device and at the edge-node level), and the promises of high scalability. Current FaaS platforms (including AWS Lambda, Google Cloud Functions, Fission, and OpenWhisk) install, manage, and monitor cloud functions while attending to operational issues such as resource auto-scaling, traffic routing, and log aggregation. Notably, the problem of supporting stateful workflows following a FaaS model in edge networks was addressed in [130]. The authors [130] focused on the problem of data transfer, which can be a performance bottleneck due to the limited speed of communication links in some edge scenarios. Distributed algorithms that implement scheduling and LB solutions must be designed and tested in edge and FC environments. The operation paradigm that usually fits the context demands that users make calls to the closer node for executing a task. As the service must be distributed among a set of nodes, the serverless computing paradigm with the FaaS is the most promising approach to use. Serverless computing [131] allows us to build and run applications and services without considering servers. Proietti and Beraldi [132] implemented a framework (called P2PFaaS) that allows the implementation of fully decentralized scheduling or LB algorithms among a set of nodes. P2FaaS is based on three basic services: (1) the scheduling service, (2) the discovery service, and (3) the learner service. P2FaaS allows the implementation of any scheduling solution, even if based on reinforcement learning (RL).

5.4. Dynamic LB Techniques in the Cloud

The cloud provides ad hoc access to shared resources such as storage, computing service, and networking facilities. This requires the involvement of optimal resource scheduling and LB policies that manage the user's load and resources [78]. Any resource scheduling and LB policies require the design of a scheduler or balancer (load), which should be able to schedule or balance the shared limited resources fairly amongst all the incoming users' requests. To manage the incoming user requests, an optimal scheduling policy comprising a scheduler should be competent enough to efficiently allocate the tasks among available VMs according to their QoS requirements (Figure 3). Static LB techniques utilize stringent policies without considering the current condition or the system's state. For their optimal implementation, these policies must know the required resources, such as the computing resources (i.e., memory, storage, and processing power capability of nodes), network parameters, communication time, the database used, etc., in advance. Although these policies are very limited, their implementation and management require low complexity from designers' and administrators' perspectives. Static policies fail to detect the load on the servers dynamically (or during the execution period). This results in unbalanced load scheduling amongst servers. This ultimately leads to unbalanced resource utilization, higher processing time, lesser availability, unnecessary offloading, migration times and cost, lower throughput, high overhead, etc. Another major problem of static policies is that they cannot dynamically adapt the load scheduling policies or guidelines since they cannot detect the most recent system's state. Therefore, static policies are unfit in distributed CC environments where the load fluctuates. Initially, static policies were implemented as *optimal* [78] and *suboptimal* [78] schemes. These schemes maintain and keep track of workload-related operations statistics in advance to reduce the waiting period of workloads. They retain the information base for all the inbound workloads and assess their execution period. Afterward, they implement the *shortest workload first* (SWF) scheduling criteria and execute the workloads accordingly. Nevertheless, there may be a high possibility that the system would get into starvation for some of the workloads during the SWF scheduling procedure. Many static LB schemes utilize the notion of round robin (RR) criteria where the incoming workloads have to be executed for a time slice and then put back into the task queue to execute another workload. Although the RR procedure is successful for web services (where the incoming requests are of identical type), it fails in CC environments, where the incoming requests are of different types altogether [78]. Dynamic policies examine the most recent system's state and make active load scheduling decisions. Furthermore, these policies allow the idea of workload migration from an overburdened VM to other underloaded VMs. Dynamic policies show flexibility considering the current condition or the system's state, which assists in maintaining adequate system performance. These policies persistently keep track of the load level of the nodes. They also monitor the resource utilization level of the nodes over a period of time, on the basis of which they decide whether the migration should be performed or not. Notably, dynamic policies interchange state and load information between nodes to estimate their workload levels. On the basis of this estimation, these policies take steps to redistribute the load among the nodes. In distributed dynamic LB schemes, all the nodes in the system participate in workload scheduling and resource provisioning. Nodes implement the notion of dispensing and reorganizing workload proficiently via an information base through which these nodes can communicate.

The dynamic LB schemes can be classified [78] on the basis of their implemented load balancer conditions: (1) general, (2) natural phenomenon-based, (3) agent-based, (4) cluster-based, and (5) workload-based.

5.4.1. General LB Schemes

The simplest way to design any LB policy for the cloud is to estimate and migrate a load of VMs. According to this concept, Ren et al. [133] introduced a generalized LB scheme for the CC environment. Their technique predicts the dynamic VM migration timing via the load assessment mechanism. This scheme utilizes a thresholding mechanism to initiate LB decisions that can prevent the issue of *peak instantaneous load triggering* (PILT). Traditionally, the trigger approach (implemented in conventional LB schemes) tends to generate excessive overhead in the system by generating frequent VM load migrations because of the PILT problem. Hence, Ren et al. [133], via their suggested scheme (load prediction), avoided such unnecessary VM migrations by identifying small PILT in the system. When the load reaches or surpasses a specified threshold, this policy predicts the future load by assessing its historical load information instead of doing a quick VM migration. This policy will only make VM migrations when at least '*k*' values exceed the threshold.

The customary VM migration techniques can be categorized as *one-threshold* (1-Th) and *dual-threshold* (du-Th) methods [133–135]. In 1-Th method-based policies, VM migration decisions are based on the upper bound threshold specified for the workloads of machines. However, two such thresholds (i.e., *upper* and *lower* bound) are specified for workloads of devices in the case of du-Th methods in which VM migrations are executed when the load surpasses the upper bound or gets below the lower bound. Similarly, power-aware resource management schemes were reported in [136,137], which utilize the notion of VM migration regulation by actively observing the level of resource utilization. Unfortunately, these threshold-based VM migration policies failed to predict the trend of arriving load and subsequently led to the issue of redundant VM migrations during the PILT problem. Liu et al. [134] considered the same issue while designing a VM migration policy by utilizing a timeseries-based workload forecasting mechanism. Their scheme utilizes the method of du-Th in which they adjusted both the upper and lower bounds (indicated for load) for reducing unnecessary VM migrations during the PILT issue and subsequently anticipated the trend of approaching loads in the cloud. However, Wang et al. [138] revealed some further issues in VM migration. They rethought the VM migration mechanisms considering migration price and latency as LB metrics. They formulated a joint optimization VM migration considering the above metrics to control the spatial deviation in the mentioned migration issue for price and latency optimization. Similar schemes [139,140] were suggested in the literature. Their strong motivation was to augment the cloud resources by considering the migration price and latency metrics. Mann [139] showed the vitality of essential factors such as the computational capacity of physical machines and the computational load of VMs during VM allocation on physical machines (PMs). The authors shed light on conventional LB methods and stated that these methods modeled these factors as single-dimensional capacities. These factors are apprehended by one number per machine. This defines VM migration as a single-dimensional issue in which a group of VMs can only be migrated or allocated on a PM when their total CPU loads are lesser than the CPU capacity of the PM. In advanced computer architecture, a VM and a PM can have many CPU cores (i.e., PM-CPU$_{\text{Core}}$ and VM-CPU$_{\text{Core}}$) [139]. Each VM-CPU$_{\text{Core}}$ can be mapped or allocated to one of the available PM-CPU$_{\text{Core}}$ during the process of VM-to-PM mapping. CPU core-level mapping information is critical in various scenarios to attain decent performance [139]. Zhao et al. [141] suggested an energy consumption and QoS guaranteed-based VM arrangement viewpoint concerning cloud setup. Notably, the energy consumption of PM should be considered when VMs are executed over them. The authors highlighted that these VMs usually share the critical physical resources of a PM when all of them execute on it. This further generates high contention among them concerning the limited sharing of physical resources, ultimately resulting in VM performance issues. Considering these issues, Zhao et al. [141] suggested an optimal VM placement procedure that handles such performance issues and the energy consumption of the PM. For this reason, they initially examined the relationship between CPU utilization and energy consumption to design a nonlinear energy model. Then, the authors developed a VM performance model to assess the performance trends of currently running VMs on a PM. Their mechanism utilizes these designed models to formulate the VM placement, effectively balancing the tradeoff between energy consumption and VM performance. Other energy-aware methodologies [142,143] aim to reduce energy consumption in clouds concerning the service providers' viewpoint. Several network (traffic) optimization mechanisms [144–146] have also been suggested to improve cloud resource utilization in compound aspects from the viewpoint of service providers. Nevertheless, Zhao et al. [141] proved that these techniques [139,140,142–147] did not consider the users' QoS viewpoint while designing VM placement schemes which may adversely affect the QoE for the user.

5.4.2. Natural Phenomenon-based LB Schemes

Workload scheduling and its balancing are NP-hard problems that cannot be solved entirely with conventional and dynamic advanced deterministic methods. Because of this, many nondeterministic methods (i.e., natural phenomenon-based metaheuristic mechanisms) have been proposed. Such methods aim to solve this NP-hard problem [148]. Hota et al. [149] reviewed LB algorithms for the CC environment. They categorized these nondeterministic techniques as metaheuristic, heuristic, and hybrid algorithms. Their survey discusses the advantages and disadvantages of these techniques (published between 2011 and 2016). However, a systematic and comprehensive review of the classified metaheuristic techniques needs to be included. Later, Milan et al. [150] classified metaheuristic techniques such as ant colony optimization (ACO), particle swarm optimization (PSO), artificial bee colony (ABC), genetic algorithm (GA), firefly optimization (FO), cuckoo search algorithm (CSA), and many others. Houssein et al. [151] classified these algorithms into swarm algorithms (i.e., ACO, PSO, and ABC) and GA algorithms and evaluated the scheduling mechanisms accordingly.

- *Swarm Algorithms*: In the case of ACO-based scheduling mechanisms, the workloads are depicted via ants. Pheromone intensity can denote the knowledge of commonly used resources and the load on each available resource. Moreover, the required critical resources in a cloud setup, i.e., VMs, can be depicted by food sources. In the case of ABC-based cloud scheduling schemes, the workloads can be demonstrated via each ABC. A beehive can be depicted as a cloud setup (environment), and VMs can be understandable as food sources. If we map the analogy of finding food for bees with cloud scheduling, this analogy can be analogous to workload scheduling on VMs in the cloud. However, assessing superior food sources can be explained as discovering and evaluating the underloaded VMs in the cloud to which all the newly arrived workloads can be scheduled. Then, PSO is adapted and mapped to handle scheduling issues in clouds. Especially in the case of PSO-based scheduling mechanisms, the number of workloads can be explained as the viewpoint of the potential solution. Each position can be depicted as a collection of candidate VMs analogous to the scheduling procedure [151].
- *Evolutionary algorithms:* The chromosomes play an essential role when dealing with GA-based mechanisms. A chromosome consists of a sequence of genes that depicts a possible solution. In such a mechanism, a basic factor, called *fitness function*, is utilized to verify whether the available chromosome is appropriate for the environment or not. Furthermore, concerning the fitness value, best-suited chromosomes are identified. Afterward, complex mutation and crossover computations are accomplished, generating more probable solutions (offspring), further creating a new population. However, the suitability of newly created possible solutions (offspring) is further verified via the fitness function in such a mechanism. This process continues until an ample number of probable solutions is found. In GA-based scheduling mechanisms, the workloads (scheduled on VMs) are depicted via the offspring computation process [151].
- *Swarm and Evolutionary Algorithms-based Load Scheduling and Balancing Schemes*: Many authors explored and revealed interesting insights into smart swarm algorithms such as ACO. By utilizing the notion of ACO, Tawfeek et al. [152] showed that the considered adapts well and suggests improved performance compared to the GA notion suggested by Dasgupta et al. [153]. Nonetheless, ACO suffers from the issue of high network overhead, and it may fall into the local optimum as revealed by Farrag et al. [154]. Considering ACO as a nature-inspired scheme, Tawfeek et al. [152] considered makespan minimization as the objective function. Their suggested mechanism utilizes a random optimization exploration method to allocate and manage the arrival workloads on VMs. Nevertheless, this technique does not address the fault tolerance factor in the system.

Fu et al. [155] highlighted the requirement of improving QoS and the competence of multidomain networks (cooperative). According to this motivation, they revealed a strict need to address the unbalanced traffic load of multi-controllers in such networks. For this reason, the authors suggested an improved ACO (IACO) scheme to handle this issue. IACO effectively addressed the controller overload and scalability issue of the control plane. IACO is a multiple-controller-based distributed architecture implemented on top of the SDN-based multi-controller architecture, which utilizes the selection probability. In IACO, such a probability is assessed through the collected topological information.

By considering the utilization of ACO to explain scheduling issues in a cloud scenario, Tsai and Rodrigues [156] insisted that numerous computational resources (e.g., CPU usage, available bandwidth, storage, and memory) can be taken into consideration at an equal time on the convergence procedure. Li et al. [157] insisted on considering the usage of processor counts on a VM, the million instructions per second (MIPS) of each processor of a VM, the estimated completion time of a workload in a VM, the network available bandwidth, and the execution time of a VM in probability estimation for the assessment of potential solutions of ACO. Moon et al. [158] proposed a novel ACO-based workload scheduling scheme utilizing the concept of 'slave ants' (i.e., SACO) in a cloud setup. SACO schedules the workloads of cloud users to VMs with an optimized factor mapping idea. The ultimate motivation of SACO is to handle the global optimization issue using the concept of slave ant. SACO utilizes dual pheromones, such as global and local, to reach the global optimum despite packing at local optima. Muteeh et al. [159] suggested a LB and scheduling-based framework (called MrLBA) for the CC environment. Their framework works on dual phases of preprocessing and optimization. Initially, MrLBA preprocesses and creates the sorted list of workloads corresponding to the available parameter information such as the completion time, dependence level amongst workloads, and amount of data to be transferred amongst workloads.

Earlier, several authors explored and revealed interesting insights into the use of the ABC notion for addressing LB issues in the cloud. The ABC notion works with three distinct categories of bees (onlooker, working, and scout) to reach the probable solutions. The working bees randomly search for solutions, and then onlooker bees utilize the optimum available solutions. Lastly, scout bees investigate the novel area in search space [160]. The authors of [161] illustrated the idea of utilizing the ABC mechanism for workload scheduling in distributed grid environment. They applied the notion of the foraging action of bees for workload scheduling on available critical limited resources. Various ABC-based LB mechanisms [123,162] schedule non-preemptive individual workloads. LD and Krishna [123] suggested an ABC-based scheme for scheduling and balancing non-preemptive independent workloads between overloaded and underloaded VMs, which aims to augment the throughput performance. This scheme manages the workloads' priorities on VMs to reduce their queuing time. Nonetheless, this scheme may suffer from the issue of scalability and single-point failures (since bees are generated from a single source). Furthermore, in this scheme, the workloads with lower priorities may suffer from starvation, resulting in more queuing time. Pan et al. [162] proposed an interaction ABC-based LB (IABC-LB) mechanism to improve production and efficiently balance the system's load. Nonetheless, their scheme does not consider the inequality of numerous load resources in a single physical machine. Hashem et al. [163] suggested another ABC-based LB mechanism that aims to disperse the task load of numerous network links to circumvent over and underusage of critically limited resources. Kruekaew and Kimpan [164] suggested an ABC-based VM scheduling strategy for the CC environment. Their objective was to reduce the makespan of processing time. The authors [165] evaluated and assessed the performance of ABC compared to GA and stated that ABC performs better than GA in some specific scenarios. Yakhchi et al. [166] proposed another CSOA-based LB scheme for efficient power management in the CC environment. Their scheme's objective is to minimize energy utilization in datacenters by blending the lowest migration time notion and server consolidation methodology. Considering the same objective, Kansal and Chana [167] suggested a firefly optimization (FO)-based power-aware VM migration policy for the cloud.

Elmagzoub et al. [168] surveyed swarm intelligence-based LB techniques in a cloud environment, while Bhargavi et al. [169] modeled the performance of some competitive swarm intelligence-based LB techniques in the cloud. Many researchers implemented new LB plans regarding the PSO mechanism. Particles' arbitrary movement is an imperative factor whenever considering PSO in the implementation of LB for any computing paradigm. The locus for every particle is recognized in every step, and the optimal particle is selected accordingly. The overall idea of this scheme is modest and has low overhead (network). Moreover, this scheme sometimes falls into a local optimum [168]. Yuan et al. [170] proposed an improved PSO (i-PSO) scheme to achieve the probable (optimal) solution within time. They offered a nature-inspired solution for optimizing VM resource scheduling by restraining particle speed in the CC environment. Subsequently, the authors revealed better results regarding the shortest average completion time and average service-level agreement (SLA) than GA. Aslanzadeh and Chaczko [171] shed light on the utilization of the endocrine scheme in designing an LB scheme for the CC environment. This scheme comprehends the concept of LB by employing the self-organization mechanism amongst overburdened VMs by enabling communication amongst them. Their method utilizes feedback in managing and transferring loads between overloaded VMs to available lightly loaded VMs. However, PSO is utilized to identify the optimal (available) lightly loaded VMs to which extra workload is assigned.

Pan and Chen [172] utilized the available LB features and proposed an improved PSO (iPSO). iPSO suggests readapting the notion of the particle's speed and position and procedures for modernizing fitness values. With the common objective of reducing makespan and augmenting resource utilization, Ebadifard and Babamir [173] suggested a static LB scheme that considers the workload as self-regulating and non-preemptive. The authors of [174] improved resource utilization and reduced makespan by considering binary version PSO, some LB, service placement, and workload assignment schemes. Miao et al. [160] focused on static assignments of VMs and dynamically postponed the LB to the arrived load. Their scheme, called adaptive Pbest discrete PSO (APDPSO), addresses the problem of conventional PSO's random particles' movement because of inadequate discretization approaches. Additionally, APDPSO addresses the issue in the *pbest* updating factor. The authors shed light that this factor in conventional techniques gets updated only bearing in mind the experience of the particles. This subsequently results in the selection of a suboptimal particle as the leader. The authors [160] suggested the constraint of sorting every particle in the presently available swarm, which ultimately assists in computing the next step of the non-dominated particle set where the particles decide their *pbest* position. Subsequently, this idea helps to estimate *pbest* more appropriately and stops the strategy from falling into local optimum.

Wang and Li [175] presented a multi-population GA (mpGA) scheme considering LB to address the workload scheduling problem in the CC environment. The mpGA's ultimate objective is to effectively address the task scheduling problem by avoiding the untimely convergence issue associated with GA. The authors insisted on using max–min and min–min algorithms for population initialization which subsequently assists mpGA in boosting search efficiency in the system. They also presented and highlighted the requirement of standardizing metropolis to screen the offspring. The motive behind such standardization was that poor individuals should not be completely ruled out; instead, they should be accepted with some probability, which consequently assists in maintaining the population diversity and dodging local optimum in the system. Their performance analysis showed that mpGA offers lower processing costs, better LB, and minimum completion time compared to the adaptive GA technique.

Cho et al. [176] suggested a hybrid technique called ACOPS by merging PSO and ACO techniques to improve VMs' LB and subsequently aimed to decrease overhead by augmenting convergence hustle. The ACOPS highlighted the requirement of adding PSO's operator to the ACO scheme to raise the effectiveness of resource scheduling. ACOPS utilizes the historical workload (request) information to estimate the incoming requests for workload dynamically and rapidly. Their performance analysis of ACOPS showed better results than the state-of-the-art approaches in terms of completion time, makespan, and response time. Their analysis also indicated that ACOPS could suggest optimum LB in a dynamic environment. Nonetheless, the presented scheme does not address fault tolerance, power consumption, and scalability. Shojafar et al. [177] suggested another hybrid LB scheme (called FUGE) that aims to accomplish optimal LB by considering cost and execution time as essential factors. FUGE indicates the fusion of GA and fuzzy theory to achieve such objectives. FUGE recommends improving the conventional GA procedure and utilizing fuzzy perceptions to augment the performance concerning makespan.

Nevertheless, FUGE may suffer from the problem of unnecessary power depletion and migration costs. Wei et al. [178] suggested another hybrid approach for workload scheduling for the cloud called simulated annealing mpGA (SAmpGA) technique. SAmpGA is the amalgamation of mpGA and the simulated annealing algorithm (SAA). The authors highlighted the requirement of standardizing metropolis for screening the offspring. They insisted on using max–min and min–min algorithms for population initialization, which subsequently assists the method in boosting search efficiency in the system. The main aim of employing SAA in the presented method is to dodge the local optimum and enhance the performance of the global optimum. The authors insisted on utilizing a family evolution mechanism based on adaptive mpGA to obtain optimal solutions and improve convergence speed. Similarly, many other hybrid algorithms [179–181] have been developed and tested over heterogeneous CC environments which suggest encouraging results regarding improvements in some essential parameters (i.e., low execution time, low makespan, better resource utilization, better LB, improved scalability, etc.). Unfortunately, until now, these policies have not been able to meet all the essential parameters. For this reason, the research community is continuously proposing changes in such algorithms.

5.4.3. Agent-Based LB Schemes

A cloud-based emergent task allocation method was created by Chen et al. [182]. The authors used the fair competitive concept and the dynamic adjustment principle to balance the load while accommodating new responsibilities. The method uses a buffer pool mechanism to improve the efficiency of the approach and a roulette wheel mechanism in which the bidder participates in the bidding process to assign the assignment to a resource. The strategies efficiently utilize resources, according to experimental findings, but they increase processing and transmission times.

Automatic resource provisioning is made easier by an application-aware LB architecture based on several software agents. Tasquier [183] introduced such an architecture that employs three agents: (1) the executor agent, (2) the provisioner agent, and (3) the monitor agent, in charge of representing running applications, managing resource scaling in and out, and keeping tabs on resource overload and underload conditions, respectively. This LB technique can assess the current level of resources and cloud elasticity. However, the author did not test the viability of his technique in a cloud setting and did not consider QoS. Gutierrez-Garcia and Ramirez-Nafarrate [184] described a collaborative agent-based LB method that uses the live VM migration concept to distribute the load among heterogeneous servers. They also suggested an agent-based LB system, which comprises an LB program to choose the initial host for VMs, strategies for VM migration, acceptance policies for VMs, and a program that recognizes the VMs that need to be migrated along with the destination. Their approach outperformed centralized LB techniques and improved task response time and resource utilization, but increased migration overhead. Keshvadi and Faghih [185] proposed a multiagent-based LB architecture that aids in maximizing resource consumption. To reduce waiting times and maintain a service-level agreement (SLA), it employs both sender- and receiver-originated approaches. The VM monitor agent (VMM agent), datacenter monitor agent (DcM agent), and negotiator ant agent (NA Agent) structure the model. The VMM agent maintains data on memory, CPU, and bandwidth utilization by VMs to track load across all supported VMs in the system. The DcM agent executes information policy utilizing data provided by the VMM agent and classifies VMs on the basis of various attributes. To learn the status of the VMs that are available in other data centers, it additionally launches NA agents.

Sangulagi and Sutagundar [186] proposed an agent-based LB technique in the sensor cloud using a neuro-fuzzy approach to achieve LB between physical nodes. Using their neuro-fuzzy approach, the redundant information is removed, and the essential information is stored in a cloud server with greater accuracy. Agents are triggered at the physical sensor network to collect the sensed information and submit it to the Cluster Head, saving the node’s energy. Their neuro-fuzzy approach balances the overall network load by rejecting redundant information generated from multiple sensor nodes and decides on the collected information. The accuracy of the decision is improved by adding weights to the decision output, which further improves system accuracy. Moreover, the network lifetime increases as a small amount information with great accuracy is saved on the cloud server. In this manner, the overall system stability through the LB approach is improved. Saini et al. [187] proposed a real-time agent-based LB algorithm for the Internet of drones in the cloud. Their algorithm decreases the communication cost of nodes/drones, speeds up the pace of LB, and improves the response and throughput of the cloud. Existing LB plans rely on the data transfer capacity or traffic condition to move the load from one node to the next. In this algorithm, a lightweight operator can move effectively starting with one node onto the next without influencing the system's load excessively.

5.4.4. Task-Based LB Schemes

Kim et al. [188] presented an LB system to process large-scale tasks using only mobile resources and no external cloud server. This plan is built on the collaborative architecture across mobile cloud computing environments. The proposed plan comprises a collaborative architecture, an agent–client architecture, and an adaptable mobile resource offloading (AMRO) architecture. It is not expected that the resources of mobile devices will always be delivered consistently in this context. Hence, an LB scheme is needed with efficient work division and optimum task allocation. A task-based LB strategy can consider the individual usage patterns and changing resource states. A technique, called task-based system load balancing (TBSLB), was created by Ramezani et al. [189]. This technique is based on the PSO algorithm to decrease the cost of VM migration. The authors also created a task transfer optimization model. They enhanced CloudSim (which has the Jswarm package incorporated in it) to assess the model proposed in [190]. Their model reduces cost, memory utilization, and VM downtime. The authors considered the utilization cost, memory downtime, and transfer time aspects. Their model can be enhanced to accommodate multi-objective PSO, more LB factors, and greater resource usage.

A task-scheduling method based on genetics and ant colonies was proposed by Wu et al. [191]. They blended the best aspects of ACO and genetic approaches. An ant colony algorithm chooses the best resource for the tasks after an earlier genetic algorithm looks for the available resources. The genetic algorithm is less efficient, is redundant, and has a longer response time. Due to a paucity of pheromones, the ant colony algorithm also performs poorly during the resource search stage. The hybrid technique enhances LB among VMs and boosts performance by incorporating elements of both algorithms.

The network-aware task placement method [192] for MapReduce decreases task completion times, total transmission times, and data cost. The challenges concerning the tasks are as follows: (1) resource availability changes dynamically as a result of releases and access over time; (2) data-fetching time for decreased tasks relies on size and location; (3) the load on the path also significantly affects data access latency. When choosing a schedule for the tasks, the load over the path should be considered to reduce data access latency. Results indicate that this method [192] increases resource utilization and reduces work completion times.

A multiple-scheduler architecture facilitates the reduction of the cost of executing massively parallel activities. Xin et al. [193] proposed the weighted random scheduling strategy to reduce task resource competition and excessive device load. Weights are allocated to the tasks on the basis of factors including cost, execution time, and communication delay. A machine with a lower cost will incur a greater cost and be more likely to be given a task to complete. To create the necessary dataset, the authors tested MATHLAB2012b's WorkFlow generator. The broader group of tasks in an interval with execution time, cost, and transmission delay was included in the datasets on which their experiments were conducted. Additionally, they considered work structure, task completion time, device dependency, and particular device set tests.

Aladwani [194] proposed the *Selecting VM with Least Load* (SVLL) LB model for task distribution. The SVLL model increases the performance of cloud systems by estimating the load of each VM and allocates the tasks for execution based on the load of the VM rather than the number of tasks allocated to the VM. This model was applied with other task-scheduling techniques such as first come, first serve and shortest task first, which results in improved total waiting time and total finish time of tasks. Elmougy et al. [195] created a task load-balancing method that combines the best elements of Round Robin (RR) scheduling and the shortest job first scheduling strategy. To balance the waiting times for the tasks, this method maintains short and long tasks in two different ready queues and employs dynamic task quantum. Throughput and starvation were considered by the authors. On the CloudSim simulator, they tested the algorithm, and the results showed that turnaround time, waiting time, and reaction time were all minimized. Additionally, extended task hunger was lessened. To balance tasks that can be upgraded in the future to reduce waiting times and the starving issue, the task quantum is not particularly effective. Alguliyev et al. [196] introduced a task-based LB model, called αPSO-TBLB. Their model offers an optimal migration of tasks causing an overload from loaded VMs to corresponding VMs in the cloud. In their optimization model, the minimization of task execution and transfer time were selected as target functions. The authors conducted simulation experiments in Cloudsim and Jswarm software tools. The simulation results proved that the suggested method provides an optimal solution for the scheduling of tasks and equal distribution of tasks to VMs. Moreover, less time consumption is obtained for the assignment process of tasks to VMs. Potluri and Rao [197] incorporated a multi-objective optimization-based VM merged technique by taking into account the precedence of tasks, LB, and fault tolerance. Using their technique, a better migration performance model was obtained to efficiently model the requirements of memory, networking, and task scheduling. Their model serves as a QoS-based resource allocation system using the fitness function to optimize execution cost, execution time, energy consumption, and task rejection ratio.

5.4.5. Cluster-Based LB Schemes

In a cloud environment, different resources are distributed throughout multiple data centers and clustered according to various factors such as server performance and storage capacity. An adaptive LB method that uses predictions assists in optimizing the utilization of data center resources. The technique [198] predicts the load in clusters; as a result, it calls back cluster resources if the workload falls below a specified threshold and adds new VMs when the burden increases. This approach reduces task response times and improves resource usage.

Daraghmi and Yuan [127] provided an LB approach to alleviate the drawbacks of a centralized controller in large data centers. With the help of designed controllers for traffic control and network management, it divides the entire network into numerous areas and assigns a controller to each region to reroute the flows. When an uneven load occurs, this LB solution migrates some of the load to the controller so that it can be managed dynamically. To manage the traffic, the authors created some solutions for the LB distributed computing technique, including greedy approaches and one distributed greedy approach. Then, they compared the outcomes with previously created controllers.

To confront the difficulties of current LB methods, Zhao et al. [199] discussed a heuristic strategy for LB based on Bayes and clustering (LB-BC). Their method produces long-term LB and is based on the Bayes theorem [200]. To select the best host, it computes the posterior probability of physical hosts and combines it with the clustering concept. The number of requested tasks, the standard deviation, and the LB effect are all factors that are taken into account. Later, the strategy was contrasted with dynamic LB, which reduces standard variation over time to a minimum. The suggested method functions only in a small region, but it can be improved for wide area networks and real-time settings.

The cluster-based task-dispatching method [201] enhances inter-cloud communication for LB in dynamic and real-time multimedia streaming. This method consists of two steps.

Using a periodic 'hello packet' broadcast to all connected neighbor servers, it first establishes the cluster for monitoring operations, managing platform complexity, and satisfying requests and satisfactory QoS for the hosts. If more than 5 tasks are in the queue that needs to be completed before the task can be started, the system decides to transfer the task requests. This approach performs better than ant colonies, while having a greater rate of task dispatching and faster reaction times. Additionally, this method can be enhanced for the real-time setting where intermediate nodes become congested owing to resource limitations, so as to decrease data loss due to congestion.

Dhurandher et al. [202] proposed a decentralized cluster-based algorithm that obtains dynamic LB in the cloud. Their algorithm supports heterogeneity and scalability and improves network congestion. Moreover, it ensures the absence of any bottleneck node due to its decentralized nature. A solution to the issue of load distribution on the nodes was proposed in [203] using a cluster-based task LB approach. By organizing tasks into clusters and distributing them among collaborating nodes, it develops a task allocation strategy by combining the principles of genetic and KUHN algorithms. The workload allocation among the data center nodes is enhanced by this technique, as is the response time. For self-scheduling techniques, Han and Chronopoulos [204] created a hierarchical distributed approach to enhance the LB and scalability of the cloud system. Both homogeneous and heterogeneous contexts can accommodate the model's operation. Four separate computation applications were used to implement the techniques in a large-scale cluster. The results demonstrate better scalability, enhanced overall performance, and decreased communication overhead. Future testing of the technique can include loops with dependencies and large-scale clusters. By clustering the load and the available resources according to specific criteria, Nishitha et al. [205] proposed an LB algorithm for cloud environments with an improvement in the throughput and a decrease in the frequency of failed task deployments.

5.5. Dynamic LB in Fog Computing

LB prevents some fog nodes from being either under- or overloaded. Recently, fog network architectures were proposed [206,207] to distribute the network traffic load in an IoT environment. Network traffic-based dynamic LB algorithms can optimize the overall network performance, while a load balancing-assisted optimized access control mechanism [206] can improve the network load conditions further. LB algorithms in fog can be classified as approximate algorithms, exact algorithms, fundamental algorithms, and hybrid algorithms.

5.5.1. Approximate Algorithms

This section discusses selected heuristic-based algorithms and probabilistic/statistic algorithms.

- *Heuristic algorithms*: These algorithms are born entirely from 'experience' with a particular optimization problem and aim to find the best solution to the problem in optimal time through 'trial and error'. Solutions in heuristic approaches may not be the best or optimal, but they can be much better than a well-informed deduction. A heuristic approach uses the details of the problem. An exact approach takes substantial time to get the optimal solution. Thus, a heuristic approach is preferable to get a near-optimal solution in the optimal time. Existing heuristic methods include hill climbing [208], Min-conflicts [209], and the analytic hierarchy process (AHP) [210].

Zahid et al. [208] suggested a framework for a three-layered architecture that consists of a distributed fog layer, a centralized cloud layer, and a consumer layer. A hill-climbing LB technique was proposed, which reduces the processing time (PT) and response time (RT) of fogs to customers. Additionally, Kamal et al. [209] presented Min-conflicts scheduling, a LB scheduling technique. The constraint fulfillment problem is solved by this algorithm using a heuristic approach. The three layers of the suggested architecture are made up of clouds, fog, and end users. Banaie et al. [210] developed multiple queuing systems to estimate the performance of a fog system to decrease the delay of data streams from

IoT devices to the applications. They used a resource caching policy and a multi-gateway architecture in the IoT sector to hasten user access to sensor data. To provide overall load fairness among the network entities, an LB strategy based on the AHP method was also used. Furthermore, Oueis et al. [211] referred to the LB problem in a fog environment to improve the quality of the user experience (QoE). They took into account many users who needed computation offloading. To handle all requests, local computation of cluster resources was required.

In IoT, many terminals possibly trigger enormous demands of bandwidth. To address this problem, Chien et al. [212] proposed a service-oriented SDN service function chain (SFC) load balance mechanism. This mechanism considers and classifies the type and priority of service required by each terminal device. Then, it adopts the heuristic algorithm to plan the transmission paths among SFCs to reduce the load of each service function (SF) and improve the overall network performance. The simulation results indicate that their approach can reduce the time of data transmission and achieve LB.

Through the resolution of the traffic routing optimization problem, Zhang and Duan [213] addressed the issue of resource allocation optimization. Their strategy creates a mathematical model for single-objective optimization. The weighted coefficient combines some sub-objectives even if the model only has one main goal. They presented an adaptive data traffic management method (a-ADTC) to address the traffic transmission optimization issue to obtain the best approximation for this optimization aim. Their heuristic greedy a-ADTC method ensures LB for a wireless network through efficient traffic movement.

– *Metaheuristic Algorithms*: A metaheuristic method is a higher-level heuristic method. It is a problem-independent method that can be practical for a wide range of problems. Any recent metaheuristic method has a diversification element and an intensification element. A balance is required between diversification and intensification to gain an influential and effective metaheuristic method. A metaheuristic method examines the entire solution space; a dissimilar set of solutions should be created. Moreover, the search must be heightened close to the neighborhood of the optimal or near-optimal solutions. Some metaheuristic algorithms include particle swarm optimization (PSO) [214–216], the fireworks algorithm [217], the bat algorithm [218], the whale optimization algorithm [219], and hybrid metaheuristics [220]. He et al. [214] proposed the fog and software-defined network (SDN). They introduced an SDN-based modified constrained optimization PSO approach for the adequate usage of the SDN and cloud/fog architecture on the Internet of vehicles. Wan et al. [215] proposed an energy-aware LB and scheduling solution based on the fog network. The authors provided an energy consumption model on the fog node that was related to the workload. Then, an optimization function was developed to balance the workload on the manufacturing cluster. The manufacturing cluster had to be prioritized as they used an upgraded PSO method to arrive at a good solution and complete tasks. Baburao et al. [216] proposed a resource allocation strategy based on PSO-based LB in a fog environment. Shi et al. [217] suggested a cloud-based mobile facial recognition architecture based on fog and the SDN architecture to solve the delay problem. In addition, they formulated LB in SDN and fog/cloud systems as an optimization problem. To solve the LB problem, they proposed the use of the fireworks algorithm (FWA) based on centralized SDN controls. Yang [218] proposed a three-layered architecture based on a fog/cloud network and big medical data. Their architecture contained the cloud, fog, and medical devices. In this architecture, their LB strategy used the bat algorithm to execute the initial setup of bat population data, which enhanced the quality of the solution in the initial sample. Malik et al. [219] proposed a fog-based framework that balances the load among fog nodes for handling the communication and processing requirements of intelligent real-time applications for patients. To provide better services to patients, they proposed an efficient cluster-based LB algorithm at the fog layer which consisted of fog nodes with various VMs. These VMs were grouped in line with their storage, functionality, computation capability, and specifications. These VMs together with the VM manager constituted a *cluster*. Such clustering assisted in the rapid allocation of tasks and reduced latency time. Concerning LB, Karthik and Kavithamani [220] proposed a whale optimization algorithm in a microgrid-connected wireless sensor network and fog settings. Moreover, Qun and Arefzadeh [221] introduced an LB method using a hybrid metaheuristic algorithm in fog-based vehicular ad hoc networks.

– *Probabilistic/Statistic Algorithms*: This section discusses LB algorithms based on probabilistic/statistic algorithms, including machine learning [222], fuzzy logic [223], game theory [224], and random walk [225].

Li et al. [222] investigated the fog infrastructure runtime features and suggested a self-similarity-based LB (SSLB) technique for large-scale fog systems. The authors introduced an adaptive threshold policy and a matching scheduling technique to ensure SSLB efficiency. Singh et al. [223] presented a load balancer based on fuzzy logic that uses different levels of tuning and the design of fuzzy control in fog environments. Their fuzzy logic model was used to perform link analysis as connections to manage traffic. Abedin, et al. [224] formulated the fog LB problem to minimize the LB cost in fog environments which are enhanced by the narrow-band Internet of things (NB-IoT). Initially, the time resource scheduling problem in NB-IoT was modeled as a bankruptcy game. Then, the transportation problem was solved by applying Vogel's approximation method of finding a viable LB solution to ensure the optimal allocation of tasks in fog environments. Lastly, Beraldi et al. [225] suggested a distributed approach for fog-LB based on the random-walk method.

5.5.2. Exact Algorithms

Optimization problems can be optimally solved using exact algorithms. In particular, each optimization problem can be solved by applying the exact search. However, larger instances require forbiddingly more time to get the optimal solution. The methodical search is significantly slower than the exact algorithms. Researchers have proposed some exact algorithms for LB in the fog environment including graph theory gradient-based, decomposition, and combinatorial.

Ningning et al. [226] developed the fog computing LB approach based on dynamic graph partitioning using the graph partitioning theory. The authors demonstrated how the fog computing framework, following cloud atomization, could flexibly design the system network and how the dynamic LB mechanism is capable of structuring the system and reducing node migration due to system modifications. Puthal et al. [227] recommended using a LB technique to assess the edge data centers (EDCs) and identify less loaded EDCs for work distribution. Finding less loaded EDCs for task distribution is easier with this method than with others. It enhances security via destination and boosts LB effectiveness. In [228], the authors suggested Dijkstra's algorithm for the LB in vehicular fog computing. Additionally, Fan and Ansari [229] presented a workload balancing model for a fog network to reduce the data flow delay in processing operations and communications by connecting IoT devices to the proper base stations.

FC was utilized by Barros et al. [230] to shorten the logical distance between the consumption site and the central distribution. IoT devices at the network edge are more effective and less expensive in managing power flow data. They assessed the effectiveness of the Newton–Raphson and Gauss–Seidel methods to create real-time computation for the load flow problem using fog. Beraldi and Alnuweiri [231] investigated LB between fog nodes and addressed the unique difficulties brought on by the fog system. They used LB techniques that were randomized and took advantage of the power of random choice. They built up sequential probing as a substitute for customary randomization processes that relied on parallel examination. The downlink of the cache-enabled fog radio access network (F-RAN) was examined by Chen and Kuehn [232], who also looked into ways to reduce power consumption and communicate more sustainably. An effective LB technique was proposed on the basis of channel states. The proposed the LB algorithm as an affordable way to create greener networks because it increases cache memory for a higher contenthitting rate. To reduce the bandwidth cost, Maswood et al. [233] introduced a mixed-integer linear programming (MILP) model in the fog/cloud environment.

Sthapit et al. [234] offered solutions for several conditions in which the cloud or fog is not present. The sensor network was first modeled using a network of queues. Then, scheduling decisions were made while taking into account LB. By employing connected automobile systems as an illustrative application, Chen et al. [235] demonstrated how vehicle mobility patterns may be used for executing periodic LB in fog servers. They provided a task model to address the scheduling issue at the server level rather than the device level. To increase serviceability in F-RANs, Dao et al. [236] presented an adaptive resource balancing (ARB) model in which resource block (RB) utilization within remote radio heads (RRHs) is balanced. This balance uses the Hungarian method and backpressure technique, taking into consideration a time-varying network topology issued by potential RRH mobility. Lastly, Mukherjee et al. [237] suggested a LB approach to specify the tradeoff between computing delays and transmission in F-RANs.

5.5.3. Fundamental Algorithms

Some studies on LB in fog computing are based on fundamental algorithms such as shortest job first, throttled, round robin (RR), and first fit. This section reviews the selected fundamental algorithms.

Nazar et al. [238] offered an LB algorithm that modified the shortest job first (MSJF) to manage the user's request load amongst VMs at the fog level. Ahmad et al. [239] also suggested an integrated cloud and fog-based platform for smart buildings' efficient energy management. For LB, the first fit (FF) method was used, which selects VMs on the basis of partitioning memory blocks. Smart buildings with several flats and IoT devices were considered in their cloud/fog-based approach. Chekired et al. [240] presented a decentralized scheduling architecture for electric vehicle (EV) energy management based on the fog system concept. Concerning particular multitenancy requirements such as latency and priority, Neto et al. [241] suggested a multitenant load distribution strategy for fog networks (MtLDF). They also provided case studies to show how their approach might be applied in contrast to a latency-driven load distribution system. Barista et al. [242] presented a method based on performing LB requirements for fog of things (FoT) platforms. They applied SDN to program IoT settings. The authors applied the FoT LB mechanism to the issues and evaluated response time and lost samples as two measures. A fog-based ecosystem was created by Tariq et al. [243] to span a huge area of six distinct parts of the planet, each of which is thought of as a separate region with several customers that submit requests on fog to acquire the required resources. For users to have a quick response with the least amount of delay, an LB strategy was put out to effectively choose VMs within a fog system. An effective LB (ELB) technique was recommended by Verma et al. [244] in addition to a fog-cloud-based architecture. Their technique maintains the data on fog networks by using information replication technology and reducing the overall need for large data centers. More studies [245–250] proposed some essential methods to provide LB in fog contexts.

5.5.4. Hybrid Algorithms

This subsection presents studies with hybrid algorithms. Hybrid algorithms apply various algorithms such as approximate, exact, and fundamental.

In fog networks, a cloud computing supplement was suggested by Naqvi et al. [251] to speed up cloud computing operations. Fog nodes employ the service broker policies to process requests on each of their 4–9VMs. Their ACO-based LB mechanism, along with throttle and RR, balances the load on VMs. Additionally, Abbasi et al. [252] focused on the application of fog computing to a smart grid (SG) that included a distributed generation environment known as a *microgrid*. The purpose of this study was to enhance resource utilization, response time, and delay time. To increase communication between customers and an energy provider, Ali et al. [253] presented a four-layered SG-based architecture. Their model covers a sizable population. Three LB algorithms were used to allocate VMs, while the service broker policies were dynamically changeable. Bin pack techniques were employed by Zubair et al. [254] as an LB mechanism together with the genetic algorithm (GA), throttle, and RR for resource allocation. In this study, an SG was combined with fog, and three locations at certain buildings and a cloud-based model were assumed. Talaat et al. [255] suggested an LB technique employing the dynamic resource allocation approach in a fog-based system based on Q-learning and GA. The LB approach manages incoming requests and distributes them across the active servers using a dynamic resource allocation mechanism while continuously tracking network traffic, gathering data on each server's load, and monitoring network traffic. A significant LB approach (ELBS) for a fog system suitable for applications in the healthcare industry was also presented by Talaat et al. [256]. With the aid of caching techniques and real-time scheduling, ELBS obtains efficient LB in a fog environment. ELBS was presented for healthcare system applications in a fog environment. Yan et al. [257] proposed a task offloading strategy based on greedy and coalitional game algorithms in a fog network. Their strategy is suitable for networks considering LB. Notably, Bali et al. [258] reviewed data offloading approaches in IoT networks at edge and fog nodes.

Avoiding overloading the fog nodes with data is crucial. On these fog nodes, data should have a brief lifespan and should be flushed or discarded regularly. A major issue with distributed systems is the possibility of data inconsistency and inaccuracy if data are removed from one node. As a result, a data flushing technique is required that can successfully flush data from fog nodes without introducing any data inconsistencies. For delay-sensitive IoT applications, Singh et al. [259] introduced a cloud/fog architecture. They also suggested an LB mechanism to efficiently distribute the workload among the fog nodes in a fog cluster. Their algorithm solves the problem of determining the optimal refresh period. Singh et al. [260] presented an energy-efficient LB algorithm, called the hybrid priority assigned laxity (HPAL) algorithm, which allocates the tasks to an appropriate VM and completes the task within the minimum time. After the task allocation, LB is handled by calculating the fog optimal time and minimum execution time. Lastly, Almutairi and Aldossary [261] investigated the effects of various edge-cloud designs on the overall IoT service time when tasks are offloaded and the effects of various application factors, such as compute and communication demands, on the overall efficiency. The loosely coupled (LC) three-tier architecture and the orchestrator-enabled (OE) three-tier architecture were used to divide the fundamental offloading strategies into two groups. Through performance-driven modeling, they further analyzed how these two approaches affect the execution of IoT services while taking into account the allocation status of computation resources and communication latency resulting from various network connections between layers.

5.5.5. LB for Distributing Incoming Traffic across a Cluster of Brokers

Message queuing telemetry transport (MQTT) [262] is an application layer protocol that is established for IoT messaging. According to MQTT design principles, network bandwidth and device resource requirements should be kept to a minimum while also aiming to assure dependability and some level of delivery assurance. Since June 2016, MQTT has been recognized by ISO as a standard (ISO/IEC 20922). The protocol continues progress by formalizing popular capability options and adding new functionalities. The most recent version, MQTT v5.0, was released in 2018. MQTT operates according to a publish/subscribe paradigm. Clients connect to a centralized broker when using MQTT, and three different participant categories exist:

- *BROKER*: An MQTT broker is a logical entity that couples publishers and subscribers. It is responsible for exchanging messages between the other participants. Widespread MQTT brokers are Mosquitto, Active-MQ, Hivemq, Bevywise, and VerneMQ. For example, HiveMQ [263] is a Java-based MQTT broker that supports MQTT 3.x and MQTT 5. Eclipse Mosquitto [264] is an open-source message broker that implements the MQTT protocol versions 5.0, 3.1.1, and 3.1. Mosquitto is lightweight and is suitable for use on all devices from low-power single-board computers to full servers.

- *PUBLISHERs*. These are the elements that send data to the broker so that it sends the data to one or more subscribers that require it.
- *SUBSCRIBERs*. These are the elements that receive data from the broker. The data they receive are the data sent by publishers.

An MQTT broker can be set up as a standalone server or as part of a cluster. A group of MQTT brokers that cooperate to form one logical MQTT broker is known as a cluster of brokers. Every broker in the cluster is operated on a different VM, typically in the cloud, and connected through a network. Broker clusters improve dependability, availability, and overall performance, since activities can be carried out in a highly parallel fashion among the cluster's brokers. Additional functionalities, such as broker discovery, failure detection, message replication, or other cluster status details, may be exposed by clustering MQTT brokers. These factors make the clustering strategy likely to use a significant number of internal resources on the computer running the broker, as well as bandwidth. To provide a single access point for all clients, load balancers are employed in conjunction with MQTT clusters. To distribute MQTT connections and device traffic among MQTT clusters, a load balancer is frequently set up in front of an MQTT cluster. When brokers are located within data servers that serve numerous customers with high message rates, this is helpful. In a cluster, the LB approach often entails connecting an incoming client to a randomly selected broker. This random-attach strategy is very simple but generates a significant amount of inter-broker traffic. Inter-broker traffic is an overhead for the system, and it increases the CPU load of the brokers, compromising the scaling behavior of the whole cluster [265].

A load balancer improves the clusters' high availability, evenly distributes the load across the cluster nodes, and enables dynamic expansion. Plain TCP connections serve as the connection between the load balancer and cluster nodes. A single MQTT cluster may support millions of customers with this configuration. MQTT clients no longer need to keep track of a list of MQTT brokers thanks to the load balancer; they just need to know one point of connection. LB commercial solutions from AWS, Aliyun, or QingCloud are supported by the EMQX MQTT broker [266]. When it comes to open-source software, HAProxy [267] can act as a load balancer for an EMQX cluster. HAProxy can also establish/terminate the TCP connections, as well as assign many dynamic scheduling policies such as RR, randomness, or least connections.

According to Detti et al. [265], a linear increase in the number of brokers making up a cluster does not always result in a corresponding linear improvement in performance. This scaling penalty can be unexpectedly large—in the range of 40%. Detti et al. [265] proposed a unique LB technique that may be applied using a greedy algorithm to address this problem and boost performance. It calls for the usage of numerous MQTT sessions per client to lessen inter-broker traffic. Through simulations and actual measurements, they demonstrated the viability and efficiency of their technique for IoT and social network applications. The scaling penalty is, thus, decreased to 10%. Kawaguchi and Bandai [268] proposed a load balancer for distributed broker environment. Their system was tested using Amazon web services and local machines. The comparative results were evaluated for several messages received and transmitted by each broker at a given time. Anwer et al. [269] considered the LB problem by investigating different MQTT-based threat models and proposing a UML profile for the effective handling of LB in IoTs employing MQTT. The intuition of the proposed profile is to introduce a lightweight extension that can provide a robust load-balanced version of the MQTT protocol. HiveMQ is a cluster-oriented solution for MQTT and is designed to make the best use of all system resources to achieve the highest message throughput. Due to this, the authors of [270] advised offloading the SSL/TLS termination onto the load balancer. The termination of SSL/TLS (secure sockets layer/transport layer security) places a heavy burden on the system CPU, which could result in a reduction in HiveMQ's message throughput. Message throughput is maintained at a consistent rate if SSL/TLS termination is offloaded to the load balancer. In the HiveMQ installations, it may occasionally be essential to reduce the cluster size. Examples of this include rolling HiveMQ upgrades or maintenance on the computers that execute HiveMQ. In these situations, it is obvious that all clients connected to the shutdown node lose their MQTT connections and typically attempt to reconnect right away (depending on the MQTT client implementation). The load balancer and the HiveMQ cluster receive a large number of connection requests as a result; this peak in network requests should be taken into account. The HiveMQ company also advises removing a scheduled shutdown node from the load balancer beforehand. Thus, the MQTT clients will disseminate. Lastly, Longo et al. [271] proposed a framework for creating and evaluating distributed architectures of MQTT brokers with realistic and customizable network topologies. Their framework is called BORDER (benchmarking framework for distributed MQTT brokers).

An in-depth study is required to identify and review more related works that focus on LB for a cluster of MQTT brokers.

5.5.6. LB Solutions for Multipath Communication in IoT

Many LB algorithms have been tested over heterogeneous IoT environments. However, these LB policies consider only various LB parameters, without considering networking parameters such as the communication overhead, link quality, loss rate, delay, and devices' energy lifetime [272]. In a real-time environment, IoT networks face the problem of monitoring sensor devices' radio resources. However, so far, few policies have addressed such an issue. Furthermore, the performance of any wireless network depends on whether the connected IoT devices are resource-enriched, as well as the current situation of the IoT network in terms of interference, collision level, channel fluctuation, bit error rate, and, most notably, the network congestion level [272–275].

In a heterogeneous IoT environment, numerous low-powered battery-driven sensors cooperate to communicate captured real-time data. In such an ad hoc networking environment, the optimal battery usage of sensors or network lifetime is one of the critical issues. Meanwhile, a denser network results in more sensor nodes communicating and more energy being consumed. This increases traffic, interference, collisions, and thus, the channel error rate. This in turn leads to retransmissions and more unnecessary power consumption, reducing the network lifetime. Considering these issues, Adil [272] suggested a power-aware forwarding mechanism to address the LB issue in IoT. The scheme manages the growing traffic in the network by scheduling it on multiple available network paths by taking care of the balance of the power consumption level. This dynamic multipath forwarding scheme is implemented at the network layer by assuming UDP as the transport layer protocol's segment. Adil et al. [273] suggested another three-phase priority-based LB scheme (enhanced-AODV) specifically for IoT. This scheme utilizes data packets classified into three network traffic classes: (1) high priority, (2) low priority, and (3) ordinary. Nevertheless, this scheme must be tested by considering TCP as a layer-4 protocol because there is an unordered reception of packets. Such packet reception generates unnecessary negative acknowledgments and buffer blocking at the receiver's end. This ultimately reduces TCP performance (due to unnecessary congestion window (cwnd) adaptations). These problems of multipath communication were addressed in [276–280].

Aljubayri et al. [281] tried to reduce MPTCP delay by utilizing the concept of opportunistic routing (OR) in an IoT environment. They presented the OR scheme for some essential multipath protocols to reduce transmissions required to send a packet to the destination and enhance the performance of IoT networks. Recently, some researchers [282–285] proposed various schemes for MPTCP while applying it to IoT networks. Pokhrel et al. [282] suggested a novel distributed transfer learning (DTL) approach to improve the performance of multipath networks in the industrial IoT (IIoT) environment. They addressed the issue of following complicated computational training practices concerning massive datasets and time/space in the case of deep learning (DL)-based network communication schemes. Although this scheme optimizes and increases learning efficiency by employing MPTCP with transfer learning (TL), the authors did not address crucial communication problems, such as out-of-order receiving, unnecessary negative acknowledgment transmissions, retransmissions, buffer blocking, and superfluous cwnd adaptations.

Dong et al. [286] suggested a novel power-aware multipath scheme by employing MPTCP with RL for heterogeneous wireless networks. Although the authors addressed the aforementioned communication problems, their method only addressed the conventional heterogeneous networks. Furthermore, considering power as an essential metric, typically for the case of limited battery-operated sensing devices in IoT networks, the authors [283,284] proposed power-aware solutions by employing MPTCP in the IoT network environment. Latest DL- and RL-based MPTCP schemes [287–290] (and surveys [291]) were considered for conventional heterogeneous, SDN and IoT networks. Xu et al. [287] addressed common communication issues by utilizing the concept of the DL approach to suggest a novel experience self-driven scheme for MPTCP. The purpose of their scheme is that the protocol should be able to adapt its functionality according to the constantly changing network environment properties. Their protocol should follow a self-adaptation procedure by continually observing and learning the characteristics of such an environment. This method suggests synchronous adaptations in all the subflow cwnds currently running on multiple available paths. Another scheme [288], called SmartCC, suggests an RL-based approach considering a heterogeneous network environment. While designing the scheme, in addition to using the subflow current state, this scheme uses dynamic states of the middle-boxes (i.e., routers). SmartCC offers encouraging results, especially in terms of throughput performance compared to conventional MPTCP. Nonetheless, accumulating dynamic state information from middle-boxes may lead to the signaling overhead and performance degradation.

6. Lessons Learnt

In this survey, we learned the following lessons:

1. There is no perfect LB technique for improving overall LB metrics. For example, some techniques account for response time, resource utilization, and migration time, while others ignore these metrics and account for others. However, some metrics appear to be mutually exclusive, e.g., depending on VM migrations, LB can lead to longer response times. Service cost is another metric that is not taken into account. Therefore, it is highly advantageous to introduce a comprehensive LB technique for improving as many metrics as possible.
2. In some situations, cloud providers need to send part of their workload to another cloud provider for LB processing. In short, using resources from multiple cloud providers is a key requirement for future LB. In this case, cloud providers face the problem of data lock-in. Our survey shows that very few articles addressed these issues. Therefore, another interesting area for future research might be to study the issue of data lock-in and cross-cloud services.
3. Even though fog computing is a hot research topic, most researchers do not yet have access to a real testbed. It was discovered that the majority of the articles employed simulator-based tools for their evaluations. Implementing the stated algorithms in the real testbed is quite difficult since the outcomes of scenarios such as scheduling in the real environment can differ from those in the simulated environment.
4. Some LB methods on FC need to be able to operate on massive scales (scalability). Some nodes, devices, and associated processes may not be guaranteed despite the small-scale validation of these approaches. Only a small number of works have addressed the scalability issue, despite its significance. Future research faces an open problem because the related publications were defined in small-scale contexts.

7. Open Research Issues

LB in IoT still has many open issues that are still waiting to be solved. For example, LB can become more efficient if it takes into account data prioritization, traffic patterns, and multi-objective optimization.

1. *Data prioritization*: Many IoT multimedia applications use real-time delay-sensitive data that require data prioritization. Data prioritization can address several issues related to enhancing QoS, video streaming, scheduling, energy, memory, security, and reducing network latency, especially in information-centric networks [292].
2. *Traffic-aware load balancing*: Through traffic identification and rerouting, an LB scheme can meet various QoS criteria by leveraging SDN's ability to monitor and operate the network. The potential of SDN switches as an LB method in machine-to-machine (M2M) networks was examined in [293]. By utilizing the advantages of rapid traffic identification and dynamic traffic rerouting in SDN, they created a traffic-aware LB system to satisfy various QoS needs of M2M traffic. They focused on LB techniques for M2M networks. However, their plan can be used for IoT networks, where machines and people communicate via IP-based networks and send their data to the cloud. In this situation, SDN can help LB by implementing flow-based networking.
3. *Multi-objective optimization in LB decisions*: There is no technique to define most QoS parameters for LB decisions in FC environments. For example, some algorithms consider only energy, cost, or response time and ignore parameters such as scalability, reliability, and security. Therefore, multi-objective optimization in LB decision making needs to be extended to consider some QoS parameters and tradeoffs between different parameters.
4. *Best solutions*: Most fog-based LB techniques (scheduling and resource allocation) fall into the NP-hard and NP-complete problem complexity categories. Several metaheuristic and heuristic algorithms have been suggested to solve them. Future research should focus on other optimization methods such as the lion optimizer algorithm [294], firefly algorithm [295], simulated annealing [296], bacterial colony optimization [297], memetic [298,299], artificial immune system [300], and grey wolf optimizer [301].
5. *Context-aware computing*: To balance the burden on fog nodes and IoT devices (which may be mobile), it is crucial to forecast where they will be in the future. Observing movement and activity patterns can help predict where nodes and devices will be in the future. LB systems can be enhanced with accessible contextual data and semantic assistance [302,303]. An exciting example of future trends is the development of LB methods for fog networks using context-aware computing.
6. *FC can enhance big data analytics*: Decision-making and recommendation systems for various smart environments have been implemented using big data analytics. Fog computing can be utilized to satisfy the requirements for big data analysis in distributed network environments that include latency, mobility, scalability, and localization. Offloading computation and data storage to nearby fog nodes in the network can achieve these metrics [304,305]. Additionally, the QoS aspects of big data analytics can be enhanced by the application of LB methods. Thus, applying LB methods for fog contexts to big data analytics can be seen as an open issue for research.
7. *Interoperability*: Fog nodes and sources are so diverse and dispersed. Thus, interoperability is a crucial success factor for LB in the IoT/fog context. Consumers often check for their favorites and other variables such as pricing and functionality because they do not want to employ just one service provider. They have the option to switch between IoT/fog-based solutions or to apply a combination of services and products to create smart LB-based IoT environments in a personalized manner due to interoperability [306]. Thus, a fascinating research direction is to consider interoperability as a crucial factor in combining the LB in IoT/fog-based services.
8. *Efficient load management in vehicular fog computing*: To enable effective cooperation through vehicle-to-vehicle and vehicle-to-infrastructure communication, an IoT-enabled cluster of cars can offer a rich reservoir of computational resources. This is feasible in vehicular fog computing, in which cars act as fog nodes for the IoT and offer cloud-like services. Then, these services are further connected with the traditional cloud to help a group of users to cooperate and perform the tasks. The dynamic nature of the vehicular ad hoc network makes efficient load management in vehicular fog

computing very challenging. Recently, Hameed et al. [307] provided a capacity-based LB approach with cluster support to carry out energy- and performance-aware vehicular fog distributed computing for effectively processing IoT tasks. Their research suggested a dynamic clustering method that forms clusters of vehicles as a function of their position, speed, and direction.

8. Conclusions

LB of workloads on VMs is a major problem in cloud computing that has drawn substantial attention from academics. LB also prevents some fog nodes in a network from being under- or overloaded. This survey compared previous related surveys on LB in the IoT. It provided a cutting-edge analysis of the problems and difficulties with LB. This study revealed that multiple LB approaches were thoroughly surveyed while taking into account various parameters. These cloud-based LB methods were classified into many groups including general LB methods, natural phenomenon-based LB, task-based LB, and agent-based LB. We talked about the benefits and drawbacks of each category of these strategies. The presented LB algorithms in fog were classified as approximate algorithms, exact algorithms, fundamental algorithms, and hybrid algorithms. Many LB algorithms take into account the majority of LB metrics and offer higher resource consumption and faster response times. However, it is necessary to enhance the methods for boosting the performance of the system.

This article can help researchers to identify research problems working in the field of LB, and it provides an overview of available load-balancing techniques.

Author Contributions: Conceptualization, D.K.; methodology, D.K. and V.K.S.; analysis and investigation, D.K. and V.K.S.; draft preparation, D.K. and V.K.S.; supervision, D.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

ABC	Artificial bee colony
ACO	Ant colony optimization
AMQP	Advanced message queuing protocol
AI	Artificial intelligence
CoAP	Constrained application protocol
CC	Cloud computing
cwnd	Congestion window
DL	Deep learning
DoDAG	Destination-oriented directed acyclic graph
DRL	Deep reinforcement learning
EC	Edge computing
EDCs	Edge data centers
FaaS	Function-as-a-service
FC	Fog computing
GA	Genetic algorithm
GSM	Global system for mobile communications
HVAC	Heating, ventilation, and air conditioning
IEEE	Institute of electrical and electronics engineers
IoT	Internet of things
IoMT	Internet of multimedia things
IP	Internet protocol
LAN	Local area network
LB	Load balancing
LPLN	Low power and lossy network

LTE	Long-term evolution
MAC	Medium access control
ML	Machine learning
MPTCP	MultiPath TCP
MQTT	Message queuing telemetry transport
M2M	Machine-to-machine
NB-API	Northbound application programming interface
PHY	Physical layer
PDR	Packet delivery ratio
PLR	Packet loss ratio
PM	Psychical machine
PSO	Particle swarm optimization
QoE	Quality of experience
QoS	Quality of service
REST	Representational state transfer protocol
RFID	Radio frequency identification device
RR	Round robin
rp-LPLNs	Routing protocol for low-power and lossy networks
RTT	Round trip time
SB-API	Southbound application programming interface
SDN	Software defined networking
SG	Smart grid
SLA	Service-level agreement
SOA	Service oriented architecture
TCP	Transmission control protocol
VM	Virtual machine
WAN	Wide area network
WSN	Wireless sensor network
5G	5th generation

References

1. Čolaković, A.; Hadžialić, M. Internet of Things (IoT): A review of enabling technologies, challenges, and open research issues. *Comput. Netw.* **2018**, *144*, 17–39. [CrossRef]
2. Li, S.; Xu, L.D.; Zhao, S. The internet of things: A survey. *Inf. Syst. Front.* **2015**, *17*, 243–259. [CrossRef]
3. Al-Fuqaha, A.; Guizani, M.; Mohammadi, M.; Aledhari, M.; Ayyash, M. Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 2347–2376. [CrossRef]
4. Alvi, S.A.; Afzal, B.; Shah, G.A.; Atzori, L.; Mahmood, W. Internet of multimedia things: Vision and challenges. *Ad Hoc Netw.* **2015**, *33*, 87–111. [CrossRef]
5. Nauman, A.; Qadri, Y.A.; Amjad, M.; Zikria, Y.B.; Afzal, M.K.; Kim, S.W. Multimedia Internet of Things: A comprehensive survey. *IEEE Access* **2020**, *8*, 8202–8250. [CrossRef]
6. Asghari, P.; Rahmani, A.M.; Javadi, H.H.S. Internet of Things applications: A systematic review. *Comput. Netw.* **2019**, *148*, 241–261. [CrossRef]
7. Hassan, R.; Qamar, F.; Hasan, M.K.; Aman, A.H.M.; Ahmed, A.S. Internet of Things and its applications: A comprehensive survey. *Symmetry* **2020**, *12*, 1674. [CrossRef]
8. Qadri, Y.A.; Nauman, A.; Zikria, Y.B.; Vasilakos, A.V.; Kim, S.W. The future of healthcare internet of things: A survey of emerging technologies. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 1121–1167. [CrossRef]
9. Talavera, J.M.; Tobón, L.E.; Gómez, J.A.; Culman, M.A.; Aranda, J.M.; Parra, D.T.; Quiroz, L.A.; Hoyos, A.; Garreta, L.E. Review of IoT applications in agro-industrial and environmental fields. *Comput. Electron. Agric.* **2017**, *142*, 283–297. [CrossRef]
10. Montori, F.; Bedogni, L.; Bononi, L. A collaborative internet of things architecture for smart cities and environmental monitoring. *IEEE Internet Things J.* **2017**, *5*, 592–605. [CrossRef]
11. Souri, A.; Rahmani, A.M.; Jafari Navimipour, N. Formal verification approaches in the web service composition: A comprehensive analysis of the current challenges for future research. *Int. J. Commun. Syst.* **2018**, *31*, e3808. [CrossRef]
12. Milani, A.S.; Navimipour, N.J. Load balancing mechanisms and techniques in the cloud environments: Systematic literature review and future trends. *J. Netw. Comput. Appl.* **2016**, *71*, 86–98. [CrossRef]
13. Zhong, H.; Fang, Y.; Cui, J. LBBSRT: An efficient SDN load balancing scheme based on server response time. *Future Gener. Comput. Syst.* **2018**, *80*, 409–416. [CrossRef]
14. Kanellopoulos, D. Congestion control for MANETs: An overview. *ICT Express* **2019**, *5*, 77–83. [CrossRef]
15. Xu, M.; Tian, W.; Buyya, R. A survey on load balancing algorithms for virtual machines placement in cloud computing. *Concurr. Comput.* **2017**, *29*, e4123. [CrossRef]

16. Nakai, A.; Madeira, E.; Buzato, L.E. On the use of resource reservation for web services load balancing. *J. Netw. Syst. Manag.* **2015**, *23*, 502–538. [CrossRef]
17. Soundarabai, P.B.; Sahai, R.K.; Thriveni, J.; Venugopal, K.R. Comparative study on load balancing techniques in distributedsystems. *Int. J. Inf. Technol.* **2012**, *6*, 53–60.
18. Al-Janabi, T.A.; Al-Raweshidy, H.S. Optimised clustering algorithm-based centralised architecture for load balancing in IoT-network. In Proceedings of the 2017 International Symposium on Wireless Communication Systems (ISWCS), Bologna, Italy, 28–31 August 2017; pp. 269–274. [CrossRef]
19. Neghabi, A.A.; Navimipour, N.J.; Hosseinzadeh, M.; Rezaee, A. Load balancing mechanisms in the software defined networks: A systematic and comprehensive review of the literature. *IEEE Access* **2018**, *6*, 14159–14178. [CrossRef]
20. Bormann, C.; Castellani, A.P.; Shelby, Z. CoAP: An application protocol for billions of tiny internet nodes. *IEEE Internet Comput.* **2012**, *16*, 62–67. [CrossRef]
21. OASIS. *OASIS Advanced Message Queuing Protocol (AMQP) Version 1.0—OASIS Standard*; OASIS: Burlington, MA, USA, 2012.
22. Kura—OSGi-Based Application Framework for M2M Service Gateways, Eclipse, Ottawa, ON, Canada. 25 September 2014. Available online: http://www.eclipse.org/proposals/technology.kura/ (accessed on 10 October 2022).
23. Ponte—M2M Bridge Framework for REST Developers, Eclipse, Ottawa, ON, USA. 25 September 2014. Available online: http://eclipse.org/proposals/technology.ponte/ (accessed on 10 October 2022).
24. Eclipse IoT. Available online: https://projects.eclipse.org/projects/iot (accessed on 10 December 2018).
25. Eclipse SCADA. Available online: http://projects.eclipse.org/projects/technology.eclipsescada (accessed on 10 December 2018).
26. Eclipse SmartHome. Available online: http://eclipse.org/proposals/technology.smarthome/ (accessed on 10 December 2018).
27. Lin, J.; Yu, W.; Zhang, N.; Yang, X.; Zhang, H.; Zhao, W. A survey on Internet of Things: Architecture, enabling technologies, security and privacy, and applications. *IEEE Internet Things J.* **2017**, *4*, 1125–1142. [CrossRef]
28. Ngu, A.H.H.; Gutierrez, M.; Metsis, V.; Nepal, S.; Sheng, M.Z. IoT middleware: A survey on issues and enabling technologies. *IEEE Internet Things J.* **2017**, *4*, 1–20. [CrossRef]
29. Qiu, T.; Chen, N.; Li, K.; Atiquzzaman, M.; Zhao, W. How can heterogeneous Internet of Things build our future: A survey. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 2011–2027. [CrossRef]
30. Atlam, H.F.; Walters, R.J.; Wills, G.B. Fog computing and the internet of things: A review. *Big Data Cogn. Comput.* **2018**, *2*, 10. [CrossRef]
31. Hu, P.; Ning, H.; Chen, L.; Daneshmand, M. An open Internet of Things system architecture based on software-defined device. *IEEE Internet Things J.* **2019**, *6*, 2583–2592. [CrossRef]
32. Wu, M.; Lu, T.J.; Ling, F.Y.; Sun, J.; Du, H.Y. Research on the architecture of Internet of Things. In Proceedings of the 2010 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE), Chengdu, China, 20–22 August 2010; pp. V5-484–V5-487. [CrossRef]
33. Ren, J.; Zhang, D.; He, S.; Zhang, Y.; Li, T. A survey on end-edge-cloud orchestrated network computing paradigms: Transparent computing, mobile edge computing, fog computing, and cloudlet. *ACM Comput. Surv.* **2019**, *52*, 1–36. [CrossRef]
34. Shafiq, D.A.; Jhanjhi, N.Z.; Abdullah, A. Load balancing techniques in cloud computing environment: A review. *J. King Saud Univ. Comput. Inf. Sci.* **2021**, *24*, 3910–3933. [CrossRef]
35. Odun-Ayo, I.; Ananya, M.; Agono, F.; Goddy-Worlu, R. Cloud computing architecture: A critical analysis. In Proceedings of the 18th International Conference on Computational Science and Applications (ICCSA), Melbourne, VIC, Australia, 2–5 July 2018; pp. 1–7. [CrossRef]
36. Adhikari, M.; Amgoth, T. Heuristic-based load-balancing algorithm for IaaS cloud. *Future Gener. Comput. Syst.* **2018**, *81*, 156–165. [CrossRef]
37. Alboaneen, D.A.; Tianfield, H.; Zhang, Y. Glowworm swarm optimisation based task scheduling for cloud computing. In Proceedings of the 2nd International Conference on Internet of Things, Data and Cloud Computing, Cambridge, UK, 22–23 March 2017; pp. 1–7. [CrossRef]
38. Ortiz, S. Software-Defined Networking: On the verge of a breakthrough? *Computer* **2013**, *46*, 10–12. [CrossRef]
39. McKeown, N.; Anderson, T.; Balakrishnan, H.; Parulkar, G.; Peterson, L.; Rexford, J.; Shenker, S.; Turner, J. OpenFlow: Enabling innovation in campus networks. *ACM SIGCOMM Comput. Commun. Rev.* **2008**, *38*, 69–74. [CrossRef]
40. Singh, M.P.; Bhandari, A. New-flow based DDoS attacks in SDN: Taxonomy, rationales, and research challenges. *Comput. Commun.* **2020**, *154*, 509–527. [CrossRef]
41. Ali, J.; Lee, G.M.; Roh, B.H.; Ryu, D.K.; Park, G. Software-Defined Networking approaches for link failure recovery: A survey. *Sustainability* **2020**, *12*, 4255. [CrossRef]
42. Latif, Z.; Sharif, K.; Li, F.; Karim, M.M.; Biswas, S.; Wang, Y. A comprehensive survey of interface protocols for software defined networks. *J. Netw. Comput. Appl.* **2020**, *156*, 102563. [CrossRef]
43. Semong, T.; Maupong, T.; Anokye, S.; Kehulakae, K.; Dimakatso, S.; Boipelo, G.; Sarefo, S. Intelligent load balancing techniques in software defined networks: A survey. *Electronics* **2020**, *9*, 1091. [CrossRef]
44. Sharma, V.K.; Verma, L.P.; Kumar, M. A fuzzy-based adaptive energy efficient load distribution scheme in ad-hoc networks. *Int. J. Intell. Syst. Appl.* **2018**, *10*, 72. [CrossRef]
45. Sharma, V.K.; Kumar, M. Adaptive energy efficient load distribution using fuzzy approach. *Adhoc Sens. Wirel. Netw.* **2017**, *39*, 123–166.

46. Sharma, V.K.; Verma, L.P.; Kumar, M.; Naha, R.K.; Mahanti, A. A-CAFDSP: An adaptive-congestion aware Fibonacci sequence based data scheduling policy. *Comput. Commun.* **2020**, *158*, 141–165. [CrossRef]
47. Sharma, V.K.; Kumar, M. Adaptive load distribution approach based on congestion control scheme in ad-hoc networks. *Int. J. Electron.* **2019**, *106*, 48–68. [CrossRef]
48. Shi, W.; Cao, J.; Zhang, Q.; Li, Y.; Xu, L. Edge computing: Vision and challenges. *IEEE Internet Things J.* **2016**, *3*, 637–646. [CrossRef]
49. Garcia Lopez, P.; Montresor, A.; Epema, D.; Datta, A.; Higashino, T.; Iamnitchi, A.; Barcellos, M.P.; Felber, P.; Riviere, E. Edge-centric computing: Vision and challenges. *ACM SIGCOMM Comput. Commun. Rev.* **2015**, *45*, 37–42. [CrossRef]
50. Beck, M.T.; Werner, M.; Feld, S.; Schimper, S. Mobile edge computing: A taxonomy. In Proceedings of the 6th International Conference on Advances in Future Internet, Lisbon, Portugal, 16–20 November 2014; pp. 48–55.
51. Hu, P.; Dhelim, S.; Ning, H.; Qiu, T. Survey on fog computing: Architecture, key technologies, applications and open issues. *J. Netw. Comput. Appl.* **2017**, *98*, 27–42. [CrossRef]
52. Kong, W.; Li, X.; Hou, L.; Li, Y. An efficient and credible multi-source trust fusion mechanism based on time decay for edge computing. *Electronics* **2020**, *9*, 502. [CrossRef]
53. Mouradian, C.; Naboulsi, D.; Yangui, S.; Glitho, R.H.; Morrow, M.J.; Polakos, P.A. A comprehensive survey on fog computing: State-of-the-art and research challenges. *IEEE Commun. Surv. Tutor.* **2017**, *20*, 416–464. [CrossRef]
54. Bonomi, F.; Milito, R.; Natarajan, P.; Zhu, J. Fog computing: A platform for internet of things and analytics. In *Big Data and Internet of Things: A Roadmap for Smart Environments*; Bessis, N., Dobre, C., Eds.; Springer: Cham, Switzerland, 2014; pp. 169–186.
55. Bonomi, F.; Milito, R.; Zhu, J.; Addepalli, S. Fog computing and its role in the internet of things. In Proceedings of the 1st Edition of the MCC Workshop on Mobile Cloud Computing, Helsinki, Finland, 17 August 2012; pp. 13–16. [CrossRef]
56. Chandak, A.; Ray, N.K. A review of load balancing in fog computing. In Proceedings of the 2019 International Conference on Information Technology (ICIT), Bhubaneswar, India, 19–21 December 2019; pp. 460–465. [CrossRef]
57. Aslanpour, M.S.; Gill, S.S.; Toosi, A.N. Performance evaluation metrics for cloud, fog and edge computing: A review, taxonomy, benchmarks and standards for future research. *Internet Things J.* **2020**, *12*, 100273. [CrossRef]
58. Sharma, V.K.; Kumar, M. Adaptive congestion control scheme in mobile ad-hoc networks. *Peer-to-Peer Netw. Appl.* **2017**, *10*, 633–657. [CrossRef]
59. Sharma, V.K.; Verma, L.P.; Kumar, M. CL-ADSP: Cross-Layer Adaptive Data Scheduling Policy in mobile ad-hoc networks. *Future Gener. Comput. Syst.* **2019**, *97*, 530–563. [CrossRef]
60. Kanellopoulos, D.; Sharma, V.K. Survey on power-aware optimization solutions for MANETs. *Electronics* **2020**, *9*, 1129. [CrossRef]
61. Song, P.; Liu, Y.; Liu, T.; Qian, D. Flow Stealer: Lightweight load balancing by stealing flows in distributed SDN controllers. *Sci. China Inf. Sci.* **2017**, *60*, 032202. [CrossRef]
62. Xu, H.; Li, X.Y.; Huang, L.; Du, Y.; Liu, Z. Partial flow statistics collection for load-balanced routing in software defined networks. *Comput. Netw.* **2017**, *122*, 43–55. [CrossRef]
63. Guo, Z.; Su, M.; Xu, Y.; Duan, Z.; Wang, L.; Hui, S.; Chao, H.J. Improving the performance of load balancing in software-defined networks through load variance-based synchronization. *Comput. Netw.* **2014**, *68*, 95–109. [CrossRef]
64. Han, T.; Ansari, N. A traffic load balancing framework for software-defined radio access networks powered by hybrid energy sources. *IEEE/ACM Trans. Netw.* **2015**, *24*, 1038–1051. [CrossRef]
65. Ghomi, E.J.; Rahmani, A.M.; Qader, N.N. Load balancing algorithms in cloud computing: A survey. *J. Netw. Comput. Appl.* **2017**, *88*, 50–71. [CrossRef]
66. Hamdan, M.; Hassan, E.; Abdelaziz, A.; Elhigazi, A.; Mohammed, B.; Khan, S.; Vasilakos, A.V.; Marsono, M.N. A comprehensive survey of load balancing techniques in Software-Defined Network. *J. Netw. Comput. Appl.* **2021**, *174*, 102856. [CrossRef]
67. Lin, Y.D.; Wang, C.C.; Lu, Y.J.; Lai, Y.C.; Yang, H.C. Two-tier dynamic load balancing in SDN-enabled Wi-Fi networks. *Wirel. Netw.* **2018**, *24*, 2811–2823. [CrossRef]
68. Boero, L.; Cello, M.; Garibotto, C.; Marchese, M.; Mongelli, M. BeaQoS: Load balancing and deadline management of queues in an OpenFlow SDN switch. *Comput. Netw.* **2016**, *106*, 161–170. [CrossRef]
69. Katyal, M.; Mishra, A. A comparative study of load balancing algorithms in cloud computing environment. *Int. J. Distrib. Cloud Comput.* **2013**, 1. [CrossRef]
70. Ghahramani, M.H.; Zhou, M.; Hon, C.T. Toward cloud computing QoS architecture: Analysis of cloud systems and cloud services. *IEEE/CAA J. Autom. Sin.* **2017**, *4*, 6–18. [CrossRef]
71. Al Nuaimi, K.; Mohamed, N.; Al Nuaimi, M.; Al-Jaroodi, J. A survey of load balancing in cloud computing: Challenges and algorithms. In Proceedings of the 2012 2nd Symposium on Network Cloud Computing and Applications, London, UK, 3–4 December 2012; pp. 137–142. [CrossRef]
72. Randles, M.; Lamb, D.; Taleb-Bendiab, A. A comparative study into distributed load balancing algorithms for cloud computing. In Proceedings of the 2010 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops, Perth, WA, Australia, 20–23 April 2010; pp. 551–556. [CrossRef]
73. Sreenivas, V.; Prathap, M.; Kemal, M. Load balancing techniques: Major challenge in cloud computing-a systematic review. In Proceedings of the 2014 International Conference on Electronic Communication Systems (ICECS), Coimbatore, India, 13–14 February 2014; pp. 1–6. [CrossRef]
74. Raghava, N.S.; Singh, D. Comparative study on load balancing techniques in cloud computing. *Open J. Mob. Comput. Cloud Comput.* **2014**, *1*, 31–42.

75. Mishra, M.; Das, A.; Kulkarni, P.; Sahoo, A. Dynamic resource management using virtual machine migrations. *IEEE Commun. Mag.* **2012**, *50*, 34–40. [CrossRef]
76. Bari, M.F.; Boutaba, R.; Esteves, R.; Granville, L.Z.; Podlesny, M.; Rabbani, M.G.; Zhang, Q.; Zhani, M.F. Data center network virtualization: A survey. *IEEE Commun. Surv. Tutor.* **2012**, *15*, 909–928. [CrossRef]
77. Thakur, A.; Goraya, M.S. A taxonomic survey on load balancing in cloud. *J. Netw. Comput. Appl.* **2017**, *98*, 43–57. [CrossRef]
78. Kumar, P.; Kumar, R. Issues and challenges of load balancing techniques in cloud computing: A survey. *ACM Comput. Surv.* **2019**, *51*, 1–35. [CrossRef]
79. Ahmad, M.O.; Khan, R.Z. Load balancing tools and techniques in cloud computing: A systematic review. In *Advances in Computer and Computational Sciences—Proceedings of the ICCCCS 2016*; Bhatia, S.K., Mishra, K.K., Tiwari, S., Singh, V.K., Eds.; Springer: Berlin/Heidelberg, Germany, 2018. [CrossRef]
80. Kaur, A.; Kaur, B.; Singh, D. Optimization techniques for resource provisioning and load balancing in cloud environment: A review. *Int. J. Inf. Eng. Electron. Bus.* **2017**, *9*, 28. [CrossRef]
81. Belgaum, M.R.; Musa, S.; Alam, M.M.; Su'ud, M.M. A systematic review of load balancing techniques in software-defined networking. *IEEE Access* **2020**, *8*, 98612–98636. [CrossRef]
82. Neghabi, A.A.; Navimipour, N.J.; Hosseinzadeh, M.; Rezaee, A. Nature-inspired meta-heuristic algorithms for solving the load balancing problem in the software-defined network. *Int. J. Commun. Syst.* **2019**, *32*, e3875. [CrossRef]
83. Li, L.; Xu, Q. Load balancing researches in SDN: A survey. In Proceedings of the 2017 7th IEEE International Conference on Electronics Information and Emergency Communication (ICEIEC), Macau, China, 21–23 July 2017; pp. 403–408. [CrossRef]
84. Zhang, J.; Yu, F.R.; Wang, S.; Huang, T.; Liu, Z.; Liu, Y. Load balancing in data center networks: A survey. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 2324–2352. [CrossRef]
85. Batista, E.; Figueiredo, G.; Prazeres, C. Load balancing between fog and cloud in fog of things based platforms through software-defined networking. *J. King Saud Univ. Comput. Inf. Sci.* **2021**, *34*, 7111–7125. [CrossRef]
86. Hui, J.W.; Culler, D.E. Extending IP to low-power, wireless personal area networks. *IEEE Internet Comput.* **2008**, *12*, 37–45. [CrossRef]
87. Pancaroglu, D.; Sen, S. Load balancing for RPL-based Internet of Things: A review. *Ad Hoc Netw.* **2021**, *116*, 102491. [CrossRef]
88. Ghaleb, B.; Al-Dubai, A.Y.; Ekonomou, E.; Alsarhan, A.; Nasser, Y.; Mackenzie, L.M.; Boukerche, A. A survey of limitations and enhancements of the ipv6 routing protocol for low-power and lossy networks: A focus on core operations. *IEEE Commun. Surv. Tutor.* **2018**, *21*, 1607–1635. [CrossRef]
89. Winter, T.; Thubert, P.; Brandt, A.; Hui, J.; Kelsey, R.; Levis, P.; Pister, K.; Struik, R.; Vasseur, J.-P.; Alexander, R. RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks (No. rfc6550). 2012. Available online: https://www.rfc-editor.org/rfc/rfc6550.html (accessed on 10 October 2022).
90. Sebastian, A.; Sivagurunathan, S. A survey on load balancing schemes in RPL based Internet of Things. *Int. J. Sci. Res. Netw. Secur. Commun.* **2018**, *6*, 43–49.
91. Kim, H.S.; Ko, J.; Culler, D.E.; Paek, J. Challenging the IPv6 routing protocol for low-power and lossy networks (RPL): A survey. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 2502–2525. [CrossRef]
92. Hou, J.; Jadhav, R.; Luo, Z. Optimization of Parent-Node Selection in RPL-Based Networks. Available online: https://datatracker.ietf.org/doc/html/draft-hou-roll-rpl-parent-selection-00 (accessed on 10 October 2022).
93. Lamaazi, H.; Benamar, N. A comprehensive survey on enhancements and limitations of the RPL protocol: A focus on the objective function. *Ad Hoc Netw.* **2020**, *96*, 102001. [CrossRef]
94. Oliveira, A.; Vazao, T. Low-power and lossy networks under mobility: A survey. *Comput. Netw.* **2016**, *107*, 339–352. [CrossRef]
95. Iova, O.; Picco, P.; Istomin, T.; Kiraly, C. Rpl: The routing standard for the internet of things... or is it? *IEEE Commun. Mag.* **2016**, *54*, 16–22. [CrossRef]
96. Zhao, M.; Kumar, A.; Joo Chong, P.H.; Lu, R. A comprehensive study of RPL and P2P-RPL routing protocols: Implementation, challenges and opportunities. *Peer-to-Peer Netw. Appl.* **2017**, *10*, 1232–1256. [CrossRef]
97. Zikria, Y.B.; Afzal, M.K.; Ishmanov, F.; Kim, S.W.; Yu, H. A survey on routing protocols supported by the Contiki Internet of Things operating system. *Future Gener. Comput. Syst.* **2018**, *82*, 200–219. [CrossRef]
98. Kim, H.S.; Kim, H.; Paek, J.; Bahk, S. Load balancing under heavy traffic in RPL routing protocol for low power and lossy networks. *IEEE Trans. Mob. Comput.* **2016**, *16*, 964–979. [CrossRef]
99. Thubert, P. *Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL)*; No. rfc6552; Internet Engineering Task Force: Fremont, CA, USA, 2012.
100. Gnawali, O.; Levis, P. *The Minimum Rank with Hysteresis Objective Function*; No. rfc6719; Internet Engineering Task Force: Fremont, CA, USA, 2012.
101. Kharrufa, H.; Al-Kashoash, H.A.; Kemp, A.H. RPL-based routing protocols in IoT applications: A review. *IEEE Sens. J.* **2019**, *19*, 5952–5967. [CrossRef]
102. Kashani, M.H.; Mahdipour, E. Load balancing algorithms in fog computing: A systematic review. *IEEE Trans. Serv. Comput.* 2022. [CrossRef]
103. Yousefpour, A.; Fung, C.; Nguyen, T.; Kadiyala, K.; Jalali, F.; Niakanlahiji, A.; Kong, J.; Jue, J.P. All one needs to know about fog computing and related edge computing paradigms: A complete survey. *J. Syst. Archit.* **2019**, *98*, 289–330. [CrossRef]

104. Yi, S.; Li, C.; Li, Q. A survey of fog computing: Concepts, applications and issues. In Proceedings of the 2015 Workshop on Mobile Big Data, New York, NY, USA, 21 June 2015; pp. 37–42. [CrossRef]
105. Xu, X.; Fu, S.; Cai, Q.; Tian, W.; Liu, W.; Dou, W.; Sun, X.; Liu, A.X. Dynamic resource allocation for load balancing in fog environment. *Wirel. Commun. Mob. Comput.* **2018**, *2018*, 6421607. [CrossRef]
106. Al-Khafajiy, M.; Baker, T.; Al-Libawy, H.; Maamar, Z.; Aloqaily, M.; Jararweh, Y. Improving fog computing performance via fog-2-fog collaboration. *Future Gener. Comput. Syst.* **2019**, *100*, 266–280. [CrossRef]
107. Memon, S.; Huang, J.; Saajid, H.; Khuda Bux, N.; Saleem, A.; Aljeroudi, Y. Novel multi-level dynamic traffic load-balancing protocol for data center. *Symmetry* **2019**, *11*, 145. [CrossRef]
108. Carlucci, G.; De Cicco, L.; Holmer, S.; Mascolo, S. Congestion control for web real-time communication. *IEEE/ACM Trans. Netw.* **2017**, *25*, 2629–2642. [CrossRef]
109. Vaquero, L.M.; Rodero-Merino, L. Finding your way in the fog: Towards a comprehensive definition of fog computing. *ACM SIGCOMM Comput. Commun. Rev.* **2014**, *44*, 27–32. [CrossRef]
110. Chiang, M.; Ha, S.; Risso, F.; Zhang, T.; Chih-Lin, I. Clarifying fog computing and networking: 10 questions and answers. *IEEE Commun. Mag.* **2017**, *55*, 18–20. [CrossRef]
111. Chiang, M.; Zhang, T. Fog and IoT: An overview of research opportunities. *IEEE Internet Things J.* **2016**, *3*, 854–864. [CrossRef]
112. Aazam, M.; Zeadally, S.; Harras, K.A. Offloading in fog computing for IoT: Review, enabling technologies, and research opportunities. *Future Gener. Comput. Syst.* **2018**, *87*, 278–289. [CrossRef]
113. Fricker, C.; Guillemin, F.; Robert, P.; Thompson, G. Analysis of an offloading scheme for data centers in the framework of fog computing. *ACM Trans. Model. Perform. Eval. Comput. Syst.* **2016**, *1*, 1–18. [CrossRef]
114. Mukherjee, M.; Shu, L.; Wang, D. Survey of fog computing: Fundamental, network applications, and research challenges. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 1826–1857. [CrossRef]
115. Ghobaei-Arani, M.; Souri, A.; Rahmanian, A.A. Resource management approaches in fog computing: A comprehensive review. *J. Grid Comput.* **2020**, *18*, 1–42. [CrossRef]
116. Hong, C.H.; Varghese, B. Resource management in fog/edge computing: A survey on architectures, infrastructure, and algorithms. *ACM Comput. Surv.* **2019**, *52*, 1–37. [CrossRef]
117. Lim, C. A survey on congestion control for RPL-based wireless sensor networks. *Sensors* **2019**, *19*, 2567. [CrossRef] [PubMed]
118. Pourghebleh, B.; Hayyolalam, V. A comprehensive and systematic review of the load balancing mechanisms in the Internet of Things. *Clust. Comput.* **2020**, *23*, 641–661. [CrossRef]
119. Gures, E.; Shayea, I.; Ergen, M.; Azmi, M.H.; El-Saleh, A.A. Machine learning based load balancing algorithms in future heterogeneous networks: A survey. *IEEE Access* **2022**, *10*, 37689–37717. [CrossRef]
120. Kaur, M.; Aron, R. A systematic study of load balancing approaches in the fog computing environment. *J. Supercomput.* **2021**, *77*, 9202–9247. [CrossRef]
121. Shahraki, A.; Taherkordi, A.; Haugen, Ø.; Eliassen, F. A survey and future directions on clustering: From WSNs to IoT and modern networking paradigms. *IEEE Trans. Netw. Serv. Manag.* **2020**, *18*, 2242–2274. [CrossRef]
122. Singh, A.; Juneja, D.; Malhotra, M. Autonomous agent based load balancing algorithm in cloud computing. *Procedia Comput. Sci.* **2015**, *45*, 832–841. [CrossRef]
123. Dinesh Babu, L.D.; Krishna, P.V. Honey bee behavior inspired load balancing of tasks in cloud computing environments. *Appl. Soft Comput.* **2013**, *13*, 2292–2303. [CrossRef]
124. Ullah, A. Artificial bee colony algorithm used for load balancing in cloud computing. *IAES Int. J. Artif. Intell.* **2019**, *8*, 156. [CrossRef]
125. Xue, H.; Kim, K.T.; Youn, H.Y. Dynamic load balancing of software-defined networking based on genetic-ant colony optimization. *Sensors* **2019**, *19*, 311. [CrossRef]
126. Panigrahi, A.; Sahu, B.; Rout, S.K.; Rath, A.K. M-Throttled: Dynamic load balancing algorithm for cloud computing. In *Intelligent and Cloud Computing—Smart Innovation, Systems and Technologies*; Mishra, D., Buyya, R., Mohapatra, P., Patnaik, S., Eds.; Springer: Singapore, 2021; Volume 194. [CrossRef]
127. Daraghmi, E.Y.; Yuan, S.M. A small world based overlay network for improving dynamic load balancing. *J. Syst. Softw.* **2015**, *107*, 187–203. [CrossRef]
128. Van Eyk, E.; Iosup, A.; Seif, S.; Thömmes, M. The SPEC cloud group's research vision on FaaS and serverless architectures. In Proceedings of the 2nd International Workshop on Serverless Computing, Las Vegas, NV, USA, 11–15 December 2017; pp. 1–4. [CrossRef]
129. Scheuner, J.; Leitner, P. Function-as-a-service performance evaluation: A multivocal literature review. *J. Syst. Softw.* **2020**, *170*, 110708. [CrossRef]
130. Cicconetti, C.; Conti, M.; Passarella, A. FaaS execution models for edge applications. *Pervasive Mob. Comput.* **2022**, *86*, 101689. [CrossRef]
131. Cassel, G.A.S.; Rodrigues, V.F.; da Rosa Righi, R.; Bez, M.R.; Nepomuceno, A.C.; da Costa, C.A. Serverless computing for Internet of Things: A systematic literature review. *Future Gener. Comput. Syst.* **2022**, *128*, 299–316. [CrossRef]
132. Mattia, G.P.; Beraldi, R. P2PFaaS: A framework for FaaS peer-to-peer scheduling and load balancing in fog and edge computing. *SSRN Electron. J.* 2022. [CrossRef]

133. Ren, H.; Lan, Y.; Yin, C. The load balancing algorithm in cloud computing environment. In Proceedings of the 2012 2nd International Conference on Computer Science and Network Technology, Changchun, China, 29–31 December 2012; pp. 925–928.
134. Liu, Y.; Gong, B.; Xing, C.; Jian, Y. A virtual machine migration strategy based on time series workload prediction using cloud model. *Math. Probl. Eng.* **2014**, *2014*, 973069. [CrossRef]
135. Chang, Y.C.; Chang, R.S.; Chuang, F.W. A predictive method for workload forecasting in the cloud environment. In *Advanced Technologies, Embedded and Multimedia for Human-Centric Computing*; Huang, Y.-M., Chao, H.-C., Deng, D.-J., Park, J.J., Eds.; Springer: Dordrecht, The Netherlands, 2014; pp. 577–585.
136. Beloglazov, A.; Abawajy, J.; Buyya, R. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Gener. Comput. Syst.* **2012**, *28*, 755–768. [CrossRef]
137. Beloglazov, A.; Buyya, R. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. *Concurr. Comput. Pract. Exp.* **2012**, *24*, 1397–1420. [CrossRef]
138. Wang, X.; Chen, X.; Yuen, C.; Wu, W.; Zhang, M.; Zhan, C. Delay-cost tradeoff for virtual machine migration in cloud data centers. *J. Netw. Comput. Appl.* **2017**, *78*, 62–72. [CrossRef]
139. Mann, Z.Á. Multicore-aware virtual machine placement in cloud data centers. *IEEE Trans. Comput.* **2016**, *65*, 3357–3369. [CrossRef]
140. Zhang, X.; Li, K.; Zhang, Y. Minimum-cost virtual machine migration strategy in datacenter. *Concurr. Comput. Pract. Exp.* **2015**, *27*, 5177–5187. [CrossRef]
141. Zhao, H.; Wang, J.; Liu, F.; Wang, Q.; Zhang, W.; Zheng, Q. Power-aware and performance-guaranteed virtual machine placement in the cloud. *IEEE Trans. Parallel Distrib. Syst.* **2018**, *29*, 1385–1400. [CrossRef]
142. De Maio, V.; Prodan, R.; Benedict, S.; Kecskemeti, G. Modelling energy consumption of network transfers and virtual machine migration. *Future Gener. Comput. Syst.* **2016**, *56*, 388–406. [CrossRef]
143. Liu, X.F.; Zhan, Z.H.; Deng, J.D.; Li, Y.; Gu, T.; Zhang, J. An energy efficient ant colony system for virtual machine placement in cloud computing. *IEEE Trans. Evol. Comput.* **2016**, *22*, 113–128. [CrossRef]
144. Belabed, D.; Secci, S.; Pujolle, G.; Medhi, D. Striking a balance between traffic engineering and energy efficiency in virtual machine placement. *IEEE Trans. Netw. Serv. Manag.* **2015**, *12*, 202–216. [CrossRef]
145. Wang, H.; Li, Y.; Zhang, Y.; Jin, D. Virtual machine migration planning in software-defined networks. *IEEE Trans. Cloud Comput.* **2017**, *7*, 1168–1182. [CrossRef]
146. Wang, R.; Wickboldt, J.A.; Esteves, R.P.; Shi, L.; Jennings, B.; Granville, L.Z. Using empirical estimates of effective bandwidth in network-aware placement of virtual machines in datacenters. *IEEE Trans. Netw. Serv. Manag.* **2016**, *13*, 267–280. [CrossRef]
147. Ilkhechi, A.R.; Korpeoglu, I.; Ulusoy, Ö. Network-aware virtual machine placement in cloud data centers with multiple traffic-intensive components. *Comput. Netw.* **2015**, *91*, 508–527. [CrossRef]
148. Remesh Babu, K.R.; Samuel, P. Enhanced bee colony algorithm for efficient load balancing and scheduling in cloud. In *Innovations in Bio-Inspired Computing and Applications*; Snasel, V., Abraham, A., Kromer, P., Pant, M., Muda, A.K., Eds.; Springer: Cham, Switzerland, 2016; pp. 67–78.
149. Hota, A.; Mohapatra, S.; Mohanty, S. Survey of different load balancing approach-based algorithms in cloud computing: A comprehensive review. In *Computational Intelligence in Data Mining*; Behera, H.S., Nayak, J., Naik, B., Abraham, A., Eds.; Springer: Berlin/Heidelberg, Germany, 2019; pp. 99–110.
150. Milan, S.T.; Rajabion, L.; Ranjbar, H.; Navimipour, N.J. Nature inspired meta-heuristic algorithms for solving the load-balancing problem in cloud environments. *Comput. Oper. Res.* **2019**, *110*, 159–187. [CrossRef]
151. Houssein, E.H.; Gad, A.G.; Wazery, Y.M.; Suganthan, P.N. Task scheduling in cloud computing based on meta-heuristics: Review, taxonomy, open challenges, and future trends. *Swarm Evol. Comput.* **2021**, *62*, 100841. [CrossRef]
152. Tawfeek, M.A.; El-Sisi, A.; Keshk, A.E.; Torkey, F.A. Cloud task scheduling based on ant colony optimization. In Proceedings of the 2013 8th International Conference on Computer Engineering & Systems (ICCES), Cairo, Egypt, 26–28 November 2013; pp. 64–69.
153. Dasgupta, K.; Mandal, B.; Dutta, P.; Mandal, J.K.; Dam, S. A genetic algorithm (GA) based load balancing strategy for cloud computing. *Procedia Technol.* **2013**, *10*, 340–347. [CrossRef]
154. Farrag, A.A.S.; Mahmoud, S.A.; El Sayed, M. Intelligent cloud algorithms for load balancing problems: A survey. In Proceedings of the 2015 IEEE 7th International Conference on Intelligent Computing and Information Systems (ICICIS), Cairo, Egypt, 12–14 December 2015; pp. 210–216.
155. Fu, Y.; Zhu, Y.; Cao, Z.; Du, Z.; Yan, G.; Du, J. Multi-Controller Load Balancing Algorithm for Test Network Based on IACO. *Symmetry* **2021**, *13*, 1901. [CrossRef]
156. Tsai, C.W.; Rodrigues, J.J. Metaheuristic scheduling for cloud: A survey. *IEEE Syst. J.* **2013**, *8*, 279–291. [CrossRef]
157. Li, K.; Xu, G.; Zhao, G.; Dong, Y.; Wang, D. Cloud task scheduling based on load balancing ant colony optimization. In Proceedings of the 2011 6th Annual ChinaGrid Conference, Liaoning, China, 22–23 August 2011; pp. 3–9. [CrossRef]
158. Moon, Y.; Yu, H.; Gil, J.M.; Lim, J. A slave ants based ant colony optimization algorithm for task scheduling in cloud computing environments. *Hum. Centric Comput. Inf. Sci.* **2017**, *7*, 28. [CrossRef]
159. Muteeh, A.; Sardaraz, M.; Tahir, M. MrLBA: Multi-resource Load Balancing Algorithm for cloud computing using ant colony optimization. *Clust. Comput.* **2021**, *24*, 3135–3145. [CrossRef]
160. Miao, Z.; Yong, P.; Mei, Y.; Quanjun, Y.; Xu, X. A discrete PSO-based static load balancing algorithm for distributed simulations in a cloud environment. *Future Gener. Comput. Syst.* **2021**, *115*, 497–516. [CrossRef]

161. Mousavinasab, Z.; Entezari-Maleki, R.; Movaghar, A. A bee colony task scheduling algorithm in computational grids. In *Digital Information Processing and Communications, Proceedings of the International Conference on Digital Information Processing and Communications (ICDIPC), Ostrava, Czech Republic, 7–9 July 2011*; Snasel, V., Platos, J., El-Qawasmeh, E., Eds.; Springer: Berlin/Heidelberg, Germany, 2011; pp. 200–210.
162. Pan, J.S.; Wang, H.; Zhao, H.; Tang, L. Interaction artificial bee colony based load balance method in cloud computing. In *Genetic and Evolutionary Computing*; Sun, H., Yang, C.-Y., Lin, C.-W., Pan, J.-S., Snasel, V., Abraham, A., Eds.; Springer: Cham, Switzerland, 2015; pp. 49–57.
163. Hashem, W.; Nashaat, H.; Rizk, R. Honey bee based load balancing in cloud computing. *KSII Trans. Internet Inf. Syst.* **2017**, *11*, 5694–5711.
164. Kruekaew, B.; Kimpan, W. Virtual machine scheduling management on cloud computing using artificial bee colony. In Proceedings of the International MultiConference of Engineers and Computer Scientists, Hong Kong, China, 12–14 March 2014; Volume 1, pp. 12–14.
165. Xu, G.; Ding, Y.; Zhao, J.; Hu, L.; Fu, X. A novel artificial bee colony approach of live virtual machine migration policy using Bayes theorem. *Sci. World J.* **2013**, *2013*, 369209. [CrossRef]
166. Yakhchi, M.; Ghafari, S.M.; Yakhchi, S.; Fazeli, M.; Patooghi, A. Proposing a load balancing method based on Cuckoo Optimization Algorithm for energy management in cloud computing infrastructures. In Proceedings of the 2015 6th International Conference on Modeling, Simulation, and Applied Optimization (ICMSAO), Istanbul, Turkey, 27–29 May 2015; pp. 1–5. [CrossRef]
167. Kansal, N.J.; Chana, I. Energy-aware virtual machine migration for cloud computing-a firefly optimization approach. *J. Grid Comput.* **2016**, *14*, 327–345. [CrossRef]
168. Elmagzoub, M.A.; Syed, D.; Shaikh, A.; Islam, N.; Alghamdi, A.; Rizwan, S. A survey of swarm intelligence based load balancing techniques in cloud computing environment. *Electronics* **2021**, *10*, 2718. [CrossRef]
169. Bhargavi, K.; Babu, B.S.; Pitt, J. Performance modeling of load balancing techniques in Cloud: Some of the recent competitive swarm artificial intelligence-based. *J. Intell. Syst.* **2021**, *30*, 40–58. [CrossRef]
170. Yuan, H.; Li, C.; Du, M. Optimal virtual machine resources scheduling based on improved particle swarm optimization in cloud computing. *J. Softw.* **2014**, *9*, 705–708. [CrossRef]
171. Aslanzadeh, S.; Chaczko, Z. Load balancing optimization in cloud computing: Applying Endocrine-particale swarm optimization. In Proceedings of the 2015 IEEE International Conference on Electro/Information Technology (EIT), DeKalb, IL, USA, 21–23 May 2015; pp. 165–169.
172. Pan, K.; Chen, J. Load balancing in cloud computing environment based on an improved particle swarm optimization. In Proceedings of the 2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 23–25 September 2015; pp. 595–598.
173. Ebadifard, F.; Babamir, S.M. A PSO-based task scheduling algorithm improved using a load-balancing technique for the cloud computing environment. *Concurr. Comput. Pract. Exp.* **2018**, *30*, e4368. [CrossRef]
174. Abbes, W.; Kechaou, Z.; Hussain, A.; Qahtani, A.M.; Almutiry, O.; Dhahri, H.; Alimi, A.M. An Enhanced Binary Particle Swarm Optimization (E-BPSO) algorithm for service placement in hybrid cloud platforms. *Neural Comput. Appl.* **2022**, 1–19. [CrossRef]
175. Wang, B.; Li, J. Load balancing task scheduling based on multi-population genetic algorithm in cloud computing. In Proceedings of the 2016 35th Chinese Control Conference (CCC), Chengdu, China, 27–29 July 2016; pp. 5261–5266.
176. Cho, K.M.; Tsai, P.W.; Tsai, C.W.; Yang, C.S. A hybrid meta-heuristic algorithm for VM scheduling with load balancing in cloud computing. *Neural Comput. Appl.* **2015**, *26*, 1297–1309. [CrossRef]
177. Shojafar, M.; Javanmardi, S.; Abolfazli, S.; Cordeschi, N. FUGE: A joint meta-heuristic approach to cloud job scheduling algorithm using fuzzy theory and a genetic method. *Clust. Comput.* **2015**, *18*, 829–844. [CrossRef]
178. Wei, X.J.; Bei, W.; Jun, L. SAMPGA task scheduling algorithm in cloud computing. In Proceedings of the 2017 36th Chinese Control Conference (CCC), Dalian, China, 26–28 July 2017; pp. 5633–5637. [CrossRef]
179. Jeyakrishnan, V.; Sengottuvelan, P. A hybrid strategy for resource allocation and load balancing in virtualized data centers using BSO algorithms. *Wirel. Pers. Commun.* **2017**, *94*, 2363–2375. [CrossRef]
180. Ben Alla, H.; Ben Alla, S.; Ezzati, A. A priority based task scheduling in cloud computing using a hybrid MCDM model. In *Ubiquitous Networking, Proceedings of the International Symposium on Ubiquitous Networking, Casablanca, Morocco, 9–12 May 2017*; Sabir, E., Armada, A.G., Ghogho, M., Debbah, M., Eds.; Springer: Cham, Switzerland, 2017; pp. 235–246.
181. Madni, S.H.H.; Abd Latiff, M.S.; Abdulhamid, S.I.M.; Ali, J. Hybrid gradient descent cuckoo search (HGDCS) algorithm for resource scheduling in IaaS cloud computing environment. *Clust. Comput.* **2019**, *22*, 301–334. [CrossRef]
182. Chen, C.; Zhu, X.; Bao, W.; Chen, L.; Sim, K.M. An agent-based emergent task allocation algorithm in clouds. In Proceedings of the 2013 IEEE 10th International Conference on High-Performance Computing and Communications, Zhangjiajie, China, 13–15 November 2013; pp. 1490–1497. [CrossRef]
183. Tasquier, L. Agent based load-balancer for multi-cloud environments. *Int. J. Cloud Comput. Res.* **2015**, *1*, 35–49.
184. Gutierrez-Garcia, J.O.; Ramirez-Nafarrate, A. Agent-based load balancing in cloud data centers. *Clust. Comput.* **2015**, *18*, 1041–1062. [CrossRef]
185. Keshvadi, S.; Faghih, B. A multi-agent based load balancing system in IaaS cloud environment. *Int. Robot. Autom. J.* **2016**, *1*, 3–8. [CrossRef]

186. Sangulagi, P.; Sutagundar, A. Agent based Load Balancing in Sensor Cloud. In Proceedings of the 2020 International Conference on Inventive Computation Technologies (ICICT), Coimbatore, India, 26–28 February 2020; pp. 342–347. [CrossRef]
187. Saini, S.; Jangra, A.; Singh, G. Real-time agent-based load-balancing algorithm for Internet of Drone (IoD) in cloud computing. In *Internet of Things*; Goyath, N., Sharma, S., Rena, A.K., Tripathi, S.L., Eds.; CRC Press: Boca Raton, FL, USA, 2022; pp. 81–94.
188. Kim, B.; Byun, H.; Heo, Y.A.; Jeong, Y.S. Adaptive job load balancing scheme on mobile cloud computing with collaborative architecture. *Symmetry* **2017**, *9*, 65. [CrossRef]
189. Ramezani, F.; Lu, J.; Hussain, F.K. Task-based system load balancing in cloud computing using particle swarm optimization. *Int. J. Parallel Program.* **2014**, *42*, 739–754. [CrossRef]
190. Calheiros, R.N.; Ranjan, R.; Beloglazov, A.; De Rose, C.A.; Buyya, R. CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw. Pract. Exp.* **2011**, *41*, 23–50. [CrossRef]
191. Wu, Z.; Xing, S.; Cai, S.; Xiao, Z.; Ming, Z. A genetic-ant-colony hybrid algorithm for task scheduling in cloud system. In *Smart Computing and Communication, Proceedings of the International Conference on Smart Computing and Communication, Shenzhen, China, 17–19 December 2016*; Qiu, M., Ed.; Springer: Cham, Switzerland, 2017; pp. 183–193. [CrossRef]
192. Shen, H.; Yu, L.; Chen, L.; Li, Z. Goodbye to fixed bandwidth reservation: Job scheduling with elastic bandwidth reservation in clouds. In Proceedings of the 2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom) Luxembourg, 12–15 December 2016; pp. 1–8. [CrossRef]
193. Xin, Y.; Xie, Z.Q.; Yang, J. A load balance oriented cost efficient scheduling method for parallel tasks. *J. Netw. Comput. Appl.* **2017**, *81*, 37–46. [CrossRef]
194. Aladwani, T. Impact of selecting virtual machine with least load on tasks scheduling algorithms in cloud computing. In Proceedings of the 2nd International Conference on Big Data, Cloud and Applications (BDCA'17), Tetouan, Morocco, 29–30 March 2017; pp. 1–7. [CrossRef]
195. Elmougy, S.; Sarhan, S.; Joundy, M. A novel hybrid of Shortest job first and round Robin with dynamic variable quantum time task scheduling technique. *J. Cloud Comput.* **2017**, *6*, 12. [CrossRef]
196. Alguliyev, R.M.; Imamverdiyev, Y.N.; Abdullayeva, F.J. PSO-based Load Balancing Method in Cloud Computing. *Autom. Control Comp. Sci.* **2019**, *53*, 45–55. [CrossRef]
197. Potluri, S.; Rao, K.S. Optimization model for QoS based task scheduling in cloud computing environment. *Indones. J. Electr. Eng. Comput. Sci.* **2020**, *18*, 1081–1088. [CrossRef]
198. Mao, Y.; Ren, D.; Chen, X. Adaptive load balancing algorithm based on prediction model in cloud computing. In Proceedings of the 2nd International Conference on Innovative Computing and Cloud Computing (ICCC'13), Wuhan, China, 1–2 December 2013; pp. 165–170. [CrossRef]
199. Zhao, J.; Yang, K.; Wei, X.; Ding, Y.; Hu, L.; Xu, G. A heuristic clustering-based task deployment approach for load balancing using Bayes theorem in cloud environment. *IEEE Trans. Parallel Distrib. Syst.* **2016**, *27*, 305–316. [CrossRef]
200. Pawlak, Z. Rough sets and decision algorithms. In *Lecture Notes in Computer Science*; Springer: Berlin/Heidelberg, Germany, 2001; Volume 2005, pp. 30–45. [CrossRef]
201. Kang, B.; Hyunseung Choo, H. A cluster-based decentralized job dispatching for the large-scale cloud. *EURASIP J. Wirel. Commun. Netw.* **2016**, *2016*, 25. [CrossRef]
202. Dhurandher, S.K.; Obaidat, M.S.; Woungang, I.; Agarwal, P.; Gupta, A.; Gupta, P. A cluster-based load balancing algorithm in cloud computing. In Proceedings of the 2014 IEEE International Conference on Communications (ICC), Sydney, Australia, 10–14 June 2014; pp. 2921–2925. [CrossRef]
203. Zegrari, F.; Idrissi, A.; Rehioui, H. Resource allocation with efficient load balancing in cloud environment. In Proceedings of the International Conference on Big Data and Advanced Wireless Technologies (BDAW'16), Blagoevgrad, Bulgaria, 10–11 November 2011; pp. 1–7. [CrossRef]
204. Han, Y.; Chronopoulos, A.T. Scalable loop self-scheduling schemes for large-scale clusters and cloud systems. *Int. J. Parallel Program.* **2017**, *45*, 595–611. [CrossRef]
205. Nishitha, M.; Hussain, S.M.; Shobha, K.R. Cluster based load balancing in cloud environment. In *ICT for Competitive Strategies*; Mishra, D.K., Dey, N., Deora, B.S., Joshi, A., Eds.; CRC Press: Boca Raton, FL, USA, 2020; pp. 731–737.
206. Agrawal, N. Dynamic load balancing assisted optimized access control mechanism for edge-fog-cloud network in Internet of Things environment. *Concurr. Comput. Pract. Exp.* **2021**, *33*, e6440. [CrossRef]
207. Naranjo, P.G.V.; Pooranian, Z.; Shojafar, M.; Conti, M.; Buyya, R. FOCAN: A Fog-supported smart city network architecture for management of applications in the Internet of Everything environments. *J. Parallel Distrib. Comput.* **2018**, *132*, 24–283. [CrossRef]
208. Zahid, M.; Javaid, N.; Ansar, K.; Hassan, K.; KaleemUllah Khan, M.; Waqas, M. Hill climbing load balancing algorithm on fog computing. In *Advances on P2P, Parallel, Grid, Cloud and Internet Computing, Proceedings of the 13th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, Taichung, Taiwan, 27–29 October 2018*; Xhafa, F., Leu, F.-Y., Ficco, M., Yang, C.-T., Eds.; Springer: Cham, Switzerland, 2019; pp. 238–251. [CrossRef]
209. Kamal, M.B.; Javaid, N.; Naqvi, S.A.A.; Butt, H.; Saif, T.; Kamal, M.D. Heuristic min-conflicts optimizing technique for load balancing on fog computing. In *Advances in Intelligent Networking and Collaborative Systems, Proceedings of the 10th International Conference on Intelligent Networking and Collaborative Systems, Bratislava, Slovakia, 5–7 September 2018*; Xhafa, F., Barolli, L., Gregus, M., Eds.; Springer: Cham, Switzerland, 2019; pp. 207–219. [CrossRef]

210. Banaie, F.; Yaghmaee, M.H.; Hosseini, S.A.; Tashtarian, F. Load-balancing algorithm for multiple gateways in Fog-based Internet of Things. *IEEE Internet Things J.* **2020**, *7*, 7043–7053. [CrossRef]
211. Oueis, J.; Strinati, E.C.; Barbarossa, S. The fog balancing: Load distribution for small cell cloud computing. In Proceedings of the 2015 IEEE 81st Vehicular Technology Conference (VTC Spring), Glasgow, UK, 11–14 May 2015; pp. 1–6. [CrossRef]
212. Chien, W.; Lai, C.; Cho, H.; Chao, H. A SDN-SFC-based service-oriented load balancing for the IoT applications. *J. Netw. Comput. Appl.* **2018**, *114*, 88–97. [CrossRef]
213. Zhang, Z.; Duan, A. An Adaptive Data Traffic Control Scheme with Load Balancing in a Wireless Network. *Symmetry* **2022**, *14*, 2164. [CrossRef]
214. He, X.; Ren, Z.; Shi, C.; Fang, J. A novel load balancing strategy of software-defined cloud/fog networking in the Internet of Vehicles. *China Commun.* **2016**, *13* (Suppl. S2), 140–149. [CrossRef]
215. Wan, J.; Chen, B.; Wang, S.; Xia, M.; Li, D.; Liu, C. Fog computing for energy-aware load balancing and scheduling in smart factory. *IEEE Trans. Ind. Inform.* **2018**, *14*, 4548–4556. [CrossRef]
216. Baburao, D.; Pavankumar, T.; Prabhu, C.S.R. Load balancing in the fog nodes using particle swarm optimization-based enhanced dynamic resource allocation method. *Appl. Nanosci.* **2021**, 1–10. [CrossRef]
217. Shi, C.; Ren, Z.; He, X. Research on load balancing for software defined cloud-fog network in real-time mobile face recognition. In *Communications and Networking, Proceedings of the International Conference on Communications and Networking in China, Chongqing, China, 24–26 September 2016*; Chen, Q., Meng, W., Zhao, L., Eds.; Springer: Cham, Switzerland, 2018; pp. 121–131.
218. Yang, J. Low-latency cloud-fog network architecture and its load balancing strategy for medical big data. *J. Ambient Intell. Humaniz. Comput.* **2020**, *11*, 1–10. [CrossRef]
219. Malik, S.; Gupta, K.; Gupta, D.; Singh, A.; Ibrahim, M.; Ortega-Mansilla, A.; Goyal, N.; Hamam, H. Intelligent load-balancing framework for fog-enabled communication in healthcare. *Electronics* **2022**, *11*, 566. [CrossRef]
220. Karthik, S.S.; Kavithamani, A. Fog computing-based deep learning model for optimization of microgrid-connected WSN with load balancing. *Wirel. Netw.* **2021**, *27*, 2719–2727. [CrossRef]
221. Qun, R.; Arefzadeh, S.M. A new energy-aware method for load balance managing in the fog-based vehicular ad hoc networks (VANET) using a hybrid optimization algorithm. *IET Commun.* **2021**, *15*, 1665–1676. [CrossRef]
222. Li, C.; Zhuang, H.; Wang, Q.; Zhou, X. SSLB: Self-Similarity-based Load Balancing for large-scale fog computing. *Arab. J. Sci. Eng.* **2018**, *43*, 7487–7498. [CrossRef]
223. Singh, S.P.; Sharma, A.; Kumar, R. Design and exploration of load balancers for fog computing using fuzzy logic. *Simul. Model. Pract. Theory* **2020**, *101*, 102017. [CrossRef]
224. Abedin, S.F.; Bairagi, A.K.; Munir, M.S.; Tran, N.H.; Hong, C.S. Fog load balancing for massive machine type communications: A game and transport theoretic approach. *IEEE Access* **2018**, *7*, 4204–4218. [CrossRef]
225. Beraldi, R.; Canali, C.; Lancellotti, R.; Proietti Mattia, G. Randomized load balancing under loosely correlated state information in fog computing. In Proceedings of the 23rd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, Alicante, Spain, 16–20 November 2020; pp. 123–127. [CrossRef]
226. Ningning, S.; Chao, G.; Xingshuo, A.; Qiang, Z. Fog computing dynamic load balancing mechanism based on graph repartitioning. *China Commun.* **2016**, *13*, 156–164. [CrossRef]
227. Puthal, D.; Obaidat, M.S.; Nanda, P.; Prasad, M.; Mohanty, S.P.; Zomaya, A.Y. Secure and sustainable load balancing of edge data centers in fog computing. *IEEE Commun. Mag.* **2018**, *56*, 60–65. [CrossRef]
228. Cui, K.; Sun, W.; Lin, B.; Sun, W. Load balancing mechanisms of unmanned surface vehicle cluster based on marine vehicular fog computing. In Proceedings of the 2020 16th International Conference on Mobility, Sensing and Networking, Tokyo, Japan, 17–19 December 2020; pp. 797–802. [CrossRef]
229. Fan, Q.; Ansari, N. Towards workload balancing in fog computing empowered IoT. *IEEE Trans. Netw. Sci. Eng.* **2018**, *7*, 253–262. [CrossRef]
230. Barros, E.; Peixoto, M.; Leite, D.; Batista, B.; Kuehne, B. A fog model for dynamic load flow analysis in smart grids. In Proceedings of the 2018 IEEE Symposium on Computers and Communications (ISCC), Natal, Brazil, 25–28 June 2018; pp. 1–6.
231. Beraldi, R.; Alnuweiri, H. Sequential randomization load balancing for fog computing. In Proceedings of the 2018 26th International Conference on Software, Telecommunications and Computer Networks (SoftCOM), Split, Croatia, 13–15 September 2018; pp. 1–6.
232. Chen, D.; Kuehn, V. Adaptive radio unit selection and load balancing in the downlink of fog radio access network. In Proceedings of the 2016 IEEE Global Communications Conference (GLOBECOM), Washington, DC, USA, 4–6 December 2016; pp. 1–7.
233. Maswood, M.M.S.; Rahman, M.R.; Alharbi, A.G.; Medhi, D. A novel strategy to achieve bandwidth cost reduction and load balancing in a cooperative three-layer fog-cloud computing environment. *IEEE Access* **2020**, *8*, 113737–113750. [CrossRef]
234. Sthapit, S.; Thompson, J.; Robertson, N.M.; Hopgood, J.R. Computational load balancing on the edge in absence of cloud and fog. *IEEE Trans. Mob. Comput.* **2018**, *18*, 1499–1512. [CrossRef]
235. Chen, Y.A.; Walters, J.P.; Crago, S.P. Load balancing for minimizing deadline misses and total runtime for connected car systems in fog computing. In Proceedings of the 2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC), Guangzhou, China, 12–15 December 2017; pp. 683–690.

236. Dao, N.N.; Lee, J.; Vu, D.N.; Paek, J.; Kim, J.; Cho, S.; Chung, K.-S.; Keum, C. Adaptive resource balancing for serviceability maximization in fog radio access networks. *IEEE Access* **2017**, *5*, 14548–14559. [CrossRef]
237. Mukherjee, M.; Liu, Y.; Lloret, J.; Guo, L.; Matam, R.; Aazam, M. Transmission and latency-aware load balancing for fog radio access networks. In Proceedings of the 2018 IEEE Global Communications Conference (GLOBECOM), Abu Dhabi, United Arab Emirates, 9–13 December 2018; pp. 1–6. [CrossRef]
238. Nazar, T.; Javaid, N.; Waheed, M.; Fatima, A.; Bano, H.; Ahmed, N. Modified shortest job first for load balancing in cloud-fog computing. In *Advances on Broadband and Wireless Computing, Communication and Applications, Proceedings of the 13th International Conference on Broadband and Wireless Computing, Communication and Applications, Taichung, Taiwan, 27–29 October 2018*; Barolli, L., Leu, F.-Y., Enokido, T., Chen, H.-C., Eds.; Springer: Cham, Switzerland, 2019; pp. 63–76.
239. Ahmad, N.; Javaid, N.; Mehmood, M.; Hayat, M.; Ullah, A.; Khan, H.A. Fog-cloud based platform for utilization of resources using load balancing technique. In *Advances in Network-Based Information Systems, Proceedings of the 21st International Conference on Network-Based Information Systems, Bratislava, Slovakia, 5–7 September 2018*; Barolli, L., Kryvinska, N., Enokido, T., Takizawa, M., Eds.; Springer: Cham, Switzerland, 2019; pp. 554–567.
240. Chekired, D.A.; Khoukhi, L.; Mouftah, H.T. Queuing model for EVs energy management: Load balancing algorithms based on decentralized fog architecture. In Proceedings of the 2018 IEEE International Conference on Communications (ICC), Kansas City, MO, USA, 20–24 May 2018; pp. 1–6. [CrossRef]
241. Neto, E.C.P.; Callou, G.; Aires, F. An algorithm to optimise the load distribution of fog environments. In Proceedings of the 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Banff, AB, Canada, 5–8 October 2017; pp. 1292–1297.
242. Batista, E.; Figueiredo, G.; Peixoto, M.; Serrano, M.; Prazeres, C. Load balancing in the fog of things platforms through software-defined networking. In Proceedings of the 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Halifax, NS, Canada, 30 July–3 August 2018; pp. 1785–1791.
243. Tariq, S.; Javaid, N.; Majeed, M.; Ahmed, F.; Nazir, S. Priority based load balancing in cloud and fog based systems. In *Advances on Broadband and Wireless Computing, Communication and Applications, Proceedings of the 13th International Conference on Broadband and Wireless Computing, Communication and Applications, Taichung, Taiwan, 27–29 October 2018*; Barolli, L., Leu, F.-Y., Enokida, T., Chen, H.-C., Eds.; Springer: Cham, Switzerland, 2019; pp. 725–736.
244. Verma, S.; Yadav, A.K.; Motwani, D.; Raw, R.S.; Singh, H.K. An efficient data replication and load balancing technique for fog computing environment. In Proceedings of the 3rd International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 16–18 March 2016; pp. 2888–2895.
245. Alqahtani, F.; Amoon, M.; Nasr, A.A. Reliable scheduling and load balancing for requests in cloud-fog computing. *Peer-to-Peer Netw. Appl.* **2021**, *14*, 1905–1916. [CrossRef]
246. Asghar, A.; Abbas, A.; Khattak, H.A.; Khan, S.U. Fog based architecture and load balancing methodology for health monitoring systems. *IEEE Access* **2021**, *9*, 96189–96200. [CrossRef]
247. Mazumdar, N.; Nag, A.; Singh, J.P. Trust-based load-offloading protocol to reduce service delays in fog-computing-empowered IoT. *Comput. Electr. Eng.* **2021**, *93*, 107223. [CrossRef]
248. Beraldi, R.; Canali, C.; Lancellotti, R.; Mattia, G.P. Distributed load balancing for heterogeneous fog computing infrastructures in smart cities. *Pervasive Mob. Comput.* **2020**, *67*, 101221. [CrossRef]
249. Rehman, A.U.; Ahmad, Z.; Jehangiri, A.I.; Ala'Anzy, M.A.; Othman, M.; Umar, A.I.; Ahmad, J. Dynamic energy efficient resource allocation strategy for load balancing in fog environment. *IEEE Access* **2020**, *8*, 199829–199839. [CrossRef]
250. Sharmin, Z.; Malik, A.W.; Rahman, A.U.; Noor, R.M. Toward sustainable micro-level fog-federated load sharing in internet of vehicles. *IEEE Internet Things J.* **2020**, *7*, 3614–3622. [CrossRef]
251. Naqvi, S.A.A.; Javaid, N.; Butt, H.; Kamal, M.B.; Hamza, A.; Kashif, M. Metaheuristic optimization technique for load balancing in cloud-fog environment integrated with smart grid. In *Advances in Network-Based Information Systems, Proceedings of the 21st International Conference on Network-Based Information Systems, Bratislava, Slovakia, 5–7 September 2018*; Barolli, L., Kryvinska, N., Enokido, T., Takizawa, M., Eds.; Springer: Cham, Switzerland, 2019; pp. 700–711.
252. Abbasi, S.H.; Javaid, N.; Ashraf, M.H.; Mehmood, M.; Naeem, M.; Rehman, M. Load stabilizing in fog computing environment using load balancing algorithm. In *Advances on Broadband and Wireless Computing, Communication and Applications Proceedings of the 13th International Conference on Broadband and Wireless Computing, Communication and Applications, Taichung, Taiwan, 27–29 October 2018*; Barolli, L., Leu, F.-Y., Enokido, T., Chen, H.-C., Eds.; Springer: Cham, Switzerland, 2019; pp. 737–750.
253. Ali, M.J.; Javaid, N.; Rehman, M.; Sharif, M.U.; Khan, M.K.; Khan, H.A. State based load balancing algorithm for smart grid energy management in fog computing. In *Advances in Intelligent Networking and Collaborative Systems, Proceedings of the 10th International Conference on Intelligent Networking and Collaborative Systems, Bratislava, Slovakia, 5–7 September 2018*; Xhafa, F., Barolli, L., Gregus, M., Eds.; Springer: Cham, Switzerland, 2019; pp. 220–232.
254. Zubair, M.; Javaid, N.; Ismail, M.; Zakria, M.; Asad Zaheer, M.; Saeed, F. Integration of cloud-fog based platform for load balancing using hybrid genetic algorithm using Bin packing technique. In *Advances on P2P, Parallel, Grid, Cloud and Internet Computing, Proceedings of the 13th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, Taichung, Taiwan, 27–29 October 2018*; Xhafa, F., Leu, F.-Y., Ficco, M., Yang, C.-T., Eds.; Springer: Cham, Switzerland, 2019; pp. 279–292. [CrossRef]
255. Talaat, F.M.; Saraya, M.S.; Saleh, A.I.; Ali, H.A.; Ali, S.H. A Load Balancing and Optimization Strategy (LBOS) using reinforcement learning in fog computing environment. *J. Ambient Intell. Humaniz. Comput.* **2020**, *11*, 4951–4966. [CrossRef]

256. Talaat, F.M.; Ali, S.H.; Saleh, A.I.; Ali, H.A. Effective load balancing strategy (ELBS) for real-time fog computing environment using fuzzy and probabilistic neural networks. *J. Netw. Syst. Manag.* **2019**, *27*, 883–929. [CrossRef]
257. Yan, J.; Wu, J.; Wu, Y.; Chen, L.; Liu, S. Task offloading algorithms for novel load balancing in homogeneous fog network. In Proceedings of the 24th International Conference on Computer Supported Cooperative Work in Design, Dalian, China, 5–7 May 2021; pp. 79–84. [CrossRef]
258. Bali, M.S.; Gupta, K.; Koundal, D.; Zaguia, A.; Mahajan, S.; Pandit, A.K. Smart architectural framework for symmetrical data offloading in IoT. *Symmetry* **2021**, *13*, 1889. [CrossRef]
259. Singh, P.; Kaur, R.; Rashid, J.; Juneja, S.; Dhiman, G.; Kim, J.; Ouaissa, M. A fog-cluster based load-balancing technique. *Sustainability* **2022**, *14*, 7961. [CrossRef]
260. Singh, S.P.; Kumar, R.; Sharma, A.; Abawajy, J.H.; Kaur, R. Energy efficient load balancing hybrid priority assigned laxity algorithm in fog computing. *Clust. Comput.* **2022**, *25*, 3325–3342. [CrossRef]
261. Almutairi, J.; Aldossary, M. Modeling and analyzing offloading strategies of IoT applications over edge computing and joint clouds. *Symmetry* **2021**, *13*, 402. [CrossRef]
262. OASIS. Message Queuing Telemetry Transport. Available online: http://mqtt.org (accessed on 10 November 2022).
263. HiveMQ GmbH. HiveMQ Community. Version: 4. 18 June 2020. Available online: https://www.hivemq.com (accessed on 18 June 2020).
264. Eclipse Mosquitto™. An Open Source MQTT Broker. Available online: https://mosquitto.org (accessed on 10 November 2022).
265. Detti, A.; Funari, L.; Blefari-Melazzi, N. Sub-linear scalability of MQTT clusters in topic-based publish-subscribe applications. *IEEE Trans. Netw. Serv. Manag.* **2020**, *17*, 1954–1968. [CrossRef]
266. EMQX: The Most Scalable MQTT Broker for IIoT. Available online: https://www.emqx.io (accessed on 10 November 2022).
267. HAProxy. Available online: https://www.haproxy.com/solutions/load-balancing/ (accessed on 10 November 2022).
268. Kawaguchi, R.; Bandai, M. A distributed MQTT Broker system for location-based IoT applications. In Proceedings of the 2019 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 11–13 January 2019; pp. 1–4. [CrossRef]
269. Anwer, H.; Azam, F.; Anwar, M.W.; Rashid, M. A model-driven approach for load-balanced MQTT protocol in internet of things (IoT). In *Complex, Intelligent, and Software Intensive Systems, Proceedings of the 13th International Conference on Complex, Intelligent, and Software Intensive Systems, Sydney, Australia, 3–5 July 2019*; Barolli, L., Hussain, F.K., Ikeda, M., Eds.; Springer: Cham, Switzerland, 2020; pp. 368–378.
270. HiveMQ. Available online: https://www.hivemq.com/docs/hivemq/4.9/user-guide/load-balancer.html#ssl-offloading (accessed on 10 October 2022).
271. Longo, E.; Redondi, A.E.C.; Cesana, M.; Manzoni, P. BORDER: A benchmarking framework for distributed MQTT brokers. *IEEE Internet Things J.* **2022**, *9*, 17728–17740. [CrossRef]
272. Adil, M. Congestion free opportunistic multipath routing load balancing scheme for Internet of Things (IoT). *Comput. Netw.* **2021**, *184*, 107707. [CrossRef]
273. Adil, M.; Song, H.; Ali, J.; Jan, M.A.; Attique, M.; Abbas, S.; Farouk, A. EnhancedAODV: A robust three phase priority-based traffic load balancing scheme for internet of things. *IEEE Internet Things J.* **2021**, *9*, 14426–14437. [CrossRef]
274. Verma, L.P.; Sharma, V.K.; Kumar, M.; Kanellopoulos, D. A novel Delay-based Adaptive Congestion Control TCP variant. *Comput. Electr. Eng.* **2022**, *101*, 108076. [CrossRef]
275. Li, Y.; Gao, Z.; Huang, L.; Du, X.; Guizani, M. Resource management for future mobile networks: Architecture and technologies. *Comput. Netw.* **2017**, *129*, 392–398. [CrossRef]
276. Verma, L.P.; Sharma, V.K.; Kumar, M.; Mahanti, A. An adaptive multi-path data transfer approach for MP-TCP. *Wirel. Netw.* **2022**, *28*, 2185–2212. [CrossRef]
277. Verma, L.P.; Sharma, V.K.; Kumar, M.; Kanellopoulos, D.; Mahanti, A. DB-CMT: A New Concurrent Multi-path Stream Control Transport Protocol. *J. Netw. Syst. Manag.* **2022**, *30*, 67. [CrossRef]
278. Tomar, P.; Kumar, G.; Verma, L.P.; Sharma, V.K.; Kanellopoulos, D.; Rawat, S.S.; Alotaibi, Y. CMT-SCTP and MPTCP Multipath Transport Protocols: A Comprehensive Review. *Electronics* **2022**, *11*, 2384. [CrossRef]
279. Verma, L.P.; Sharma, V.K.; Kumar, M. New delay-based fast retransmission policy for CMT-SCTP. *Int. J. Intell. Syst. Appl.* **2018**, *10*, 59–66. [CrossRef]
280. Hurtig, P.; Grinnemo, K.J.; Brunstrom, A.; Ferlin, S.; Alay, Ö.; Kuhn, N. Low-latency scheduling in MPTCP. *IEEE/ACM Trans. Netw.* **2018**, *27*, 302–315. [CrossRef]
281. Aljubayri, M.; Peng, T.; Shikh-Bahaei, M. Reduce delay of multipath TCP in IoT networks. *Wirel. Netw.* **2021**, *27*, 4189–4198. [CrossRef]
282. Pokhrel, S.R.; Pan, L.; Kumar, N.; Doss, R.; Vu, H.L. Multipath TCP meets transfer learning: A novel edge-based learning for industrial IoT. *IEEE Internet Things J.* **2021**, *8*, 10299–10307. [CrossRef]
283. Morawski, M.; Ignaciuk, P. A green multipath TCP framework for industrial internet of things applications. *Comput. Netw.* **2021**, *187*, 107831. [CrossRef]
284. Dong, Z.; Cao, Y.; Xiong, N.; Dong, P. EE-MPTCP: An Energy-Efficient Multipath TCP Scheduler for IoT-based power grid monitoring systems. *Electronics* **2022**, *11*, 3104. [CrossRef]
285. Silva, C.F.; Ferlin, S.; Alay, O.; Brunstrom, A.; Kimura, B.Y. IoT traffic offloading with MultiPath TCP. *IEEE Commun. Mag.* **2021**, *59*, 51–57. [CrossRef]

286. Dong, P.; Shen, R.; Wang, Q.; Zhang, D.; Li, Y.; Zuo, Y.; Yang, W.; Zhang, L. Multipath TCP meets Reinforcement Learning: A novel energy-efficient scheduling approach in heterogeneous wireless networks. *IEEE Wirel. Commun.* **2022**, 1–9. [CrossRef]
287. Xu, Z.; Tang, J.; Yin, C.; Wang, Y.; Xue, G. Experience-driven congestion control: When multi-path TCP meets deep reinforcement learning. *IEEE J. Sel. Areas Commun.* **2019**, *37*, 1325–1336. [CrossRef]
288. Li, W.; Zhang, H.; Gao, S.; Xue, C.; Wang, X.; Lu, S. SmartCC: A reinforcement learning approach for multipath TCP congestion control in heterogeneous networks. *IEEE J. Sel. Areas Commun.* **2019**, *37*, 2621–2633. [CrossRef]
289. Naeem, F.; Srivastava, G.; Tariq, M. A software defined network based fuzzy normalized neural adaptive multipath congestion control for the internet of things. *IEEE Trans. Netw. Sci. Eng.* **2020**, *7*, 2155–2164. [CrossRef]
290. Chen, Y.R.; Rezapour, A.; Tzeng, W.G.; Tsai, S.C. Rl-routing: An SDN routing algorithm based on deep reinforcement learning. *IEEE Trans. Netw. Sci. Eng.* **2020**, *7*, 3185–3199. [CrossRef]
291. Tang, F.; Mao, B.; Kawamoto, Y.; Kato, N. Survey on machine learning for intelligent end-to-end communication toward 6G: From network access, routing to traffic control and streaming adaption. *IEEE Commun. Surv. Tutor.* **2021**, *23*, 1578–1598. [CrossRef]
292. Zahedinia, M.S.; Khayyambashi, M.R.; Bohlooli, A. Fog-based caching mechanism for IoT data in information centric network using prioritization. *Comput. Netw.* **2022**, *213*, 109082. [CrossRef]
293. Chen, Y.J.; Wang, L.C.; Chen, M.C.; Huang, P.M.; Chung, P.J. SDN-enabled traffic-aware load balancing for M2M networks. *IEEE Internet Things J.* **2018**, *5*, 1797–1806. [CrossRef]
294. Yazdani, M.; Jolai, F. Lion Optimization Algorithm (LOA): A nature-inspired metaheuristic algorithm. *J. Comput. Des. Eng.* **2016**, *3*, 24–36. [CrossRef]
295. Yang, X.S.; Slowik, A. Firefly algorithm. In *Swarm Intelligence Algorithms*; Yang, X.-S., Slowik, A., Eds.; CRC Press: Boca Raton, FL, USA, 2020; pp. 163–174.
296. Hanine, M.; Benlahmar, E.H. A load-balancing approach using an improved simulated annealing algorithm. *J. Inf. Process. Syst.* **2020**, *16*, 132–144. [CrossRef]
297. Niu, B.; Wang, H. Bacterial colony optimization. *Discret. Dyn. Nat. Soc.* **2012**, *2012*, 698057. [CrossRef]
298. Keshanchi, B.; Navimipour, N.J. Priority-based task scheduling in the cloud systems using a memetic algorithm. *J. Circuits Syst. Comput.* **2016**, *25*, 1650119. [CrossRef]
299. Dowlatshahi, M.B.; Rafsanjani, M.K.; Gupta, B.B. An energy aware grouping memetic algorithm to schedule the sensing activity in WSNs-based IoT for smart cities. *Appl. Soft Comput.* **2021**, *108*, 107473. [CrossRef]
300. Dasgupta, D. Advances in artificial immune systems. *IEEE Comput. Intell. Mag.* **2006**, *1*, 40–49. [CrossRef]
301. Faris, H.; Aljarah, I.; Al-Betar, M.A.; Mirjalili, S. Grey wolf optimizer: A review of recent variants and applications. *Neural Comput. Appl.* **2018**, *30*, 413–435. [CrossRef]
302. Minh, Q.T.; Kamioka, E.; Yamada, S. CFC-ITS: Context-aware fog computing for intelligent transportation systems. *IT Prof.* **2018**, *20*, 35–45. [CrossRef]
303. Islam, M.S.U.; Kumar, A.; Hu, Y.C. Context-aware scheduling in fog computing: A survey, taxonomy, challenges and future directions. *J. Netw. Comput. Appl.* **2021**, *180*, 103008. [CrossRef]
304. Zhang, W.; Zhang, Z.; Chao, H.C. Cooperative fog computing for dealing with big data in the internet of vehicles: Architecture and hierarchical resource management. *IEEE Commun. Mag.* **2017**, *55*, 60–67. [CrossRef]
305. Badidi, E.; Mahrez, Z.; Sabir, E. Fog computing for smart cities' big data management and analytics: A review. *Future Internet* **2020**, *12*, 190. [CrossRef]
306. Negash, B.; Westerlund, T.; Tenhunen, H. Towards an interoperable Internet of Things through a web of virtual things at the Fog layer. *Future Gener. Comput. Syst.* **2019**, *91*, 96–107. [CrossRef]
307. Hameed, A.R.; ul Islam, S.; Ahmad, I.; Munir, K. Energy-and performance-aware load-balancing in vehicular fog computing. *Sustain. Comput. Inform. Syst.* **2021**, *30*, 100454. [CrossRef]