Hindawi
Journal of Electrical and Computer Engineering
Volume 2017, Article ID 2363240, 11 pages
https://doi.org/10.1155/2017/2363240

Research Article

Leveraging Fog Computing for Scalable IoT Datacenter Using Spine-Leaf Network Topology

K. C. Okafor,[1] Ifeyinwa E. Achumba,[2] Gloria A. Chukwudebe,[2] and Gordon C. Ononiwu[1]

[1]*Department of Mechatronics Engineering, Federal University of Technology Owerri, Ihiagwa, Nigeria*
[2]*Department of Electrical and Electronic Engineering, Federal University of Technology Owerri, Ihiagwa, Nigeria*

Correspondence should be addressed to K. C. Okafor; kennedy.okafor@futo.edu.ng

Received 13 July 2016; Accepted 2 April 2017; Published 24 April 2017

Academic Editor: Raj Senani

With the Internet of Everything (IoE) paradigm that gathers almost every object online, huge traffic workload, bandwidth, security, and latency issues remain a concern for IoT users in today's world. Besides, the scalability requirements found in the current IoT data processing (in the cloud) can hardly be used for applications such as assisted living systems, Big Data analytic solutions, and smart embedded applications. This paper proposes an extended cloud IoT model that optimizes bandwidth while allowing edge devices (Internet-connected objects/devices) to smartly process data without relying on a cloud network. Its integration with a massively scaled spine-leaf (SL) network topology is highlighted. This is contrasted with a legacy multitier layered architecture housing network services and routing policies. The perspective offered in this paper explains how low-latency and bandwidth intensive applications can transfer data to the cloud (and then back to the edge application) without impacting QoS performance. Consequently, a spine-leaf Fog computing network (SL-FCN) is presented for reducing latency and network congestion issues in a highly distributed and multilayer virtualized IoT datacenter environment. This approach is cost-effective as it maximizes bandwidth while maintaining redundancy and resiliency against failures in mission critical applications.

1. Introduction

Scalability is a desirable feature of a disruptive technology such as the Internet of Things (IoT) and IoE. Ideally, the cloud computing foundations for today's IoT/IoE paradigm have opened up technology perspectives and applications for growing enterprises and their services. IoT is simply defined as the network of physical objects or "things" embedded with sensor electronics and IPv6 connectivity to enable valuable and service oriented exchange of data with a vendor platform, or even other connected devices. This can be achieved through advanced protocols requiring absence of human control.

With today's IoT, it is possible to bring consumer electronic devices including home appliances such as medical devices, fridges, cameras, and sensors into the Internet environment [1]. Machine-to-machine communication which enables "everything" connectivity to the Internet network is not only a reality but also an integral part of day-to-day living and interactions. With IoT, disruptive applications such as smart cities/vibrant ecosystems, smart grid, governance/knowledge-driven platforms, and agricultural and health systems can be repositioned to offer reliable Quality of Service (QoS). For instance, using IoT, intelligent transport system (ITS) applications can monitor city traffic 24/7 using a wireless sensor video surveillance system and then send the gathered information to the users on their smart mobile devices via a global positioning system (GPS) transceiver. This could alert users to avoid traffic jams and prevent accidents.

Interestingly, IoT essentially supports layered integration, real-time data transfer, and analytics of data generated by smart embedded devices (data streams). These will improve the quality of life, enhance urbanization, facilitate efficient health care delivery, and handle natural disasters among other things. In the layered integration, the data plane of the Fog layer enables computing services to be housed at the edge of the network as opposed to servers in a legacy datacenter. For application purposes, the integration framework in context emphasizes proximity to end users. It creates even distribution of local resources, reduces latency for QoS, and facilitates edge stream processing. The overall benefits are availability, consolidated user experience, resilience, and redundancy. This makes the application of IoE paradigm widely accepted and used on a real-time basis. With the layered integration concept discussed in Section 3.2, smart devices, wearable health monitoring devices, connected vehicles, and augmented reality can optimally fit into ISO/IEC 20248 standards which deal with general data aggregation in IoT.

However, Fog IoT model is fundamentally built into cloud datacenters [2–4]. These cloud datacenter structures are classified into two major categories. The first is the switch-centric datacenter, which organizes switches into structures other than trees and puts the interconnection intelligence on switches. Some notable examples are Fat-Tree [5], VL2 [6], PortLand [7], Dragonfly [8], and PERCS [9]. The second category is server-centric datacenter, which leverages the rapid growth scale of the server hardware including its multiple Network Interface Card (NIC) ports to put the interconnection and routing intelligence on the servers principally. Examples are DCell [10], FiConn [11], BCube [12], and BCN [13]. The other types are the containerized datacenters [14–17].

For these datacenters, their environment is mainly used to process, store, and analyze large volumes of data on demand. Also, with the cloud datacenters, hosting of IoT applications, storage of a large volume of data, and execution of live data analytics require a robust architecture. Similarly, within a typical cloud datacenter, a large number of servers are found interconnected by network devices using a specific networking structure. These networks have high performance switches which serve as an interface connecting and sharing the data in a distributed fashion. But, in a typical IoT transaction, network bandwidth congestion can arise when large volumes of data are moved to edge node or cluster in the cloud datacenter environment. This will normally violate service level agreement (SLA). Obviously, most cloud datacenters offer scalability via redundancy and resilience. However, as virtualization, cloud computing, and distributed cloud computing become increasingly popular in the IoT datacenters, there is an urgent need to evolve well balanced network support for IoT services.

It is obvious that the traditional datacenter network having the core, aggregation, and access model performs well for north-south traffic, that is, traffic that travels in and out of the datacenter. However, HTTP/S web service, exchange, FTP, and e-mail services require a lot of remote client/server communication. In this case, their network architecture is normally designed for core redundancy and resiliency against outages or failure. More so, in a production scenario, about 50% of the critical network link path is blocked by the spanning-tree protocol (STP). This is in order to prevent event based network loops. As such, these paths constitute redundant backup wasting about 50% maximum bandwidth.

In this paper, with the possibility of latency issues (data offloading), wastage of bandwidth, and network outage as a result of saturated STP in these traditional networks, a better alternative is considered. In this case, a Fog computing network (FCN) for scalable IoT datacenter is proposed. This is based on a spine-leaf network topology. The use of this type of computing would relieve enormous real-time workloads, reduce latency issues, and make for smarter responses as more people use IoT applications and services. This is because the Fog DC is the most useful in facilitating Big Data and real-time analytics, while delivering and moving data closer to the user. With Fog DC, location awareness and global aggregation for the edge devices are made possible for IoE. The major components in its integration layer are the Fog data plane having its typical instances:

(i) Real-time pooling of end user idle computing (e.g., storage/bandwidth resources)

(ii) Content caching at the edge and bandwidth management services

(iii) Client-driven distributed broadcast

(iv) Client-to-client direct communications (e.g., WiMAX, LTE 3/4G, and Wi-Fi)

(v) Cloudlets with mini datacenters

The second component is the control plane which has the following instances:

(i) Smart Over-the-Top (SOTT) content management procedure

(ii) Fog-driven radio access network, for example, Radio Network Controllers (RNC)

(iii) Client-based protocol controls from the edge

(iv) Client-controlled cloud storage from the edge

(v) Session management at the edge

(vi) Ubiquitous crowd sensing of network states at the edge

(vii) Edge analytics and real-time stream processing (data mining) from the edge

Furthermore, the isolated benefits of Fog DC include the following:

(i) Real-time processing and cyberphysical system control especially in tactile Internet and edge data analytics, as well as interfacing between humans and objects

(ii) Intelligence and cognition awareness for end-to-end communication via edge/client devices

(iii) Network efficiency via pooling of local resources by objects at the edge

(iv) Scalability and agility which make for faster and cheaper computation at the client and edge devices

(v) Security via encrypted and multipath traffic in the end-to-end network system

With the above background, this paper is now organized as follows. In Section 2, a review of classical works in scalable computing, applications, and services is presented. Also, scalable IoT networks are studied while highlighting their limitations. In Section 3, a framework for building a cost-effective, fault-tolerant, and symmetrical Fog spine-leaf network structure for IoT datacenter is presented. In this regard, a Fog computing system architecture, including its architectural framework, and the IoT requirements are discussed. In Section 4, the design of the proposed scalable IoT network using spine-leaf topology is presented. In addition, the merits and limitations of the network are presented. Section 5 discusses the design implementation, integration techniques, and SL-FCN interfaces. Section 6 presents the performance evaluation while focusing on simulation case studies for scalable IoT networks. Conclusion and future works are discussed in Section 7.

2. Related Works

To facilitate scalability in IoT based datacenter networks, various schemes have been proposed in the literature. This section will look at scalable IoT networks and the overall research gaps of the traditional computing DCNs.

2.1. Scalable Strategies to IoT Networks. Meng et al. [18] proposed a Software Defined (SD) approach to network virtualization in cloud datacenters. By optimizing the placement of VMs on host machines, traffic patterns among VMs can be better aligned with the communication distance between them. The work used traffic traces collected from production datacenters to evaluate their proposed VM placement algorithm while showing performance improvement compared to existing generic methods that do not take advantage of traffic patterns and datacenter network characteristics. Wells et al. [19] proposed a Mixed-Mode Multicore (MMM) system to support such changing requirements in a virtual cloud network. In this network, certain applications (or portions of applications) run in high performance mode using a single core, while other applications (including the system software) run in a highly reliable mode using Dual-Modular Redundancy (DMR) [20]. These are considered as fault-tolerant schemes. Wang et al. [21] proposed a joint optimization strategy for achieving energy efficiency of datacenter networks. This was done by proposing a unified optimization framework. In their framework, the work considered taking advantage of the application characteristics and topology features to integrate virtual machine assignment and traffic engineering. Under this framework, two algorithms were proposed for assigning virtual machines and routing traffic flows, respectively. However, emphasis was excluded from Fog computing.

Wang and Ng [22] highlighted that most cloud service providers use machine virtualization strategies to provide flexible and cost-effective resource sharing among users. Such scalability as found in Amazon EC2 [23] and GoGrid [24] uses Xen virtualization [25] to support multiple virtual machine instances on a single physical server. The virtual machine instances share physical processors and I/O interfaces with other instances achieving some form of scalability. Virtualization was obviously identified as a scalable strategy that impacts the computation and communication performance of IoT cloud services. For instance, Xen [25] represents an open-source x86 virtual machine monitor which can create multiple virtual machines on a physical machine running enterprise cloud service. In this regard, each virtual machine runs an operating system instance while using a scheduler which runs in the Xen hypervisor to schedule virtual machines on the processors.

Besides, Huang and Peng [26] proposed a novel model NetCloud of data placement and query for cloud computing in DCN based on the DCell datacenter design model. The work analyzed an efficient, fault-tolerant, self-organizing data placement and query model NetCloud based on DCell datacenter design architecture. Other works on fault-tolerant and scalable datacenter networks were carried out in [27–32]. Interestingly, most of the works on scalable computing networks provide full real-time visibility of both physical and virtual infrastructure.

2.2. Research Gaps. It was observed that some cloud scalable networks lack key capabilities such as multihypervisor support, integrated security, end-to-end mapping for IoT application placement, and ease of maintenance.

(i) Again, the software network virtualization strategy treats physical and virtual infrastructure as separate entities and denies end users the ability to manage computer resources as well as allowing for QoS monitoring and management.

(ii) Traditional datacenter tree-like structures previously enumerated have a variety of challenges, such as limited server-to-server connectivity, vulnerability to single point of failure, lack of agility, insufficient scalability, and smart resource fragmentation. To achieve a scalable IoT network with low-latency (response time) performance and fault recovery under variable data streams, there is a need to support adaptive, scalable load balancing and elastic runtime scaling of cloud reducers. This has the capacity of taking care of workload variation on the datacenter system.

(iii) Also, there is a need to develop a low-latency and fault-tolerant mechanism that has minimal overhead during regular operations. Hence, real-time parallel fault recovery is vital. This paper has a perspective that all these limitations of traditional cloud computing will adversely affect scalable IoT design.

3. Fog Computing System Architecture

3.1. Architectural Framework. Fog computing is a distributed computing paradigm that extends the services provided by the cloud to the edge of the network [1]. With Fog grid, this enables seamless fusion of cloud and edge resources for efficient data offloading in a short time. It offers efficient resource provisioning, management, and programming of computing, networking, and storage services between datacenters and edge end devices. Essentially, this type of computing essentially involves wireless data transfer to distributed devices

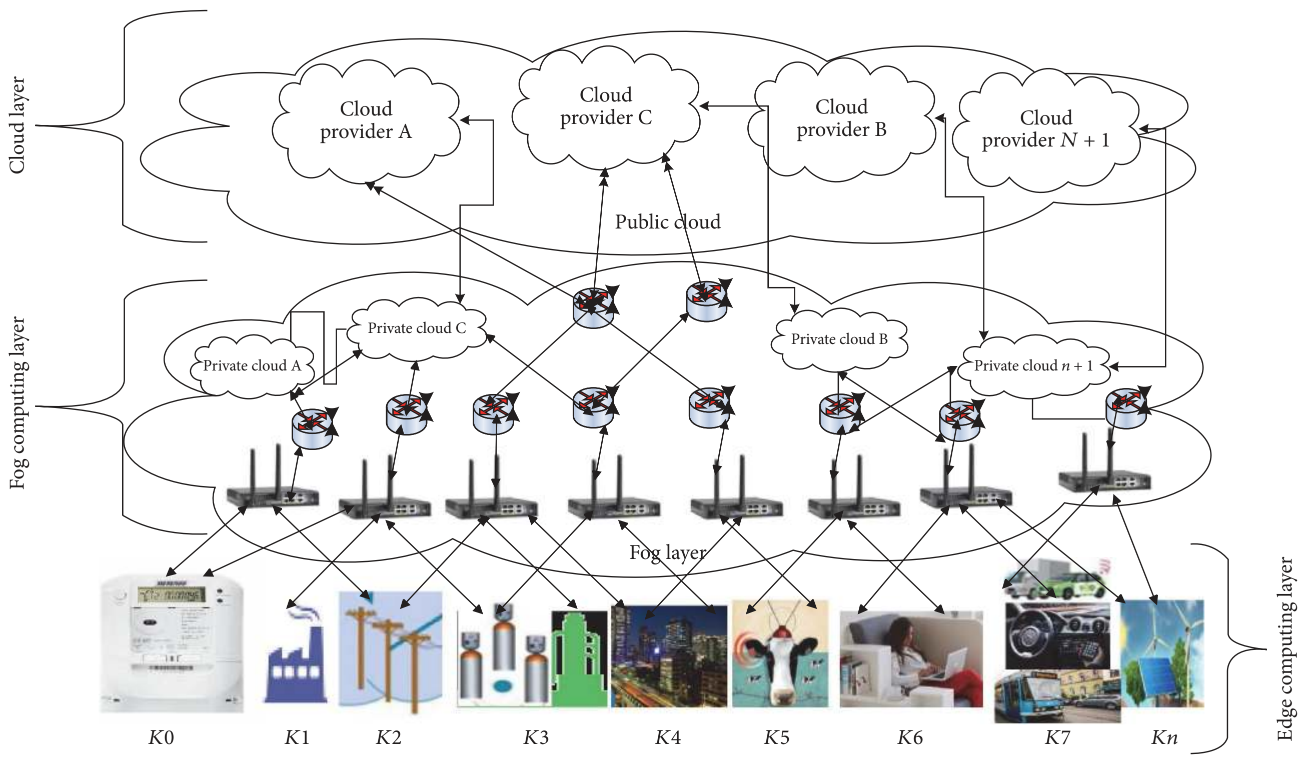

Figure 1: A conceptual framework for Fog distributed data processing (the author's model with Visio 2013).

in the Internet of Things (IoT) network cloud. Unlike with cloud computing where there is an on-demand network use of a shared pool of configurable computing resources (such as networks, servers, storage, applications, and services) that is usually provisioned with minimal vendor management efforts, Fog computing has its application services and components running on both cloud and end devices via its smart gateways and routers.

Figure 1 illustrates distributed data processing in a Fog computing environment. With the Fog layer, edge computing devices can seamlessly connect to the federated cloud to facilitate data offloading from the cloud. In this case, computing is dynamically distributed across the cloud sites. The network elements for scalability and QoS can be determined. With Fog computing, there is no need for storage renting infrastructure or even renting of computing services and applications. Rather, the Fog layer is optimized to support Internet of things (IoT) and Internet of Everything (IoE) smarter mobility, seamless resource and interface heterogeneity, handshake with the cloud, and distributed data analytics. This is to address requirements of IoT/IoE applications that require low latency with a wide and dense geographical distribution [1]. In context, this computing concept leverages both edge and cloud computing. In other words, it uses edge devices for close proximity to the endpoints (edge) and also leverages the on-demand scalability of cloud resources.

With the established framework, novel network integration for the emerging Internet of Everything (IoE) applications (K_1-K_n) is expedient. This is because these applications have serious demand for real-time and predictable latency (e.g., industrial automation, agriculture, renewable energy, transportation, and networks of sensors and actuators). The network depends on the wide geographical distribution Fog system model for real-time Big Data and real-time analytics. Invariably, this will support remote and densely distributed data collection points, thereby enhancing critical Big Data dimensions, that is, volume, variety, and velocity, as the fourth axis. In this paper, the conceptual framework has support for application enabled platforms, management and automation, Fog data stream computing, physical and cybersecurity components, data analytics, and cloud network connectivity. A layered architecture is presented in Section 3.2.

3.2. IoT Requirements in Fog Computing Architecture. In deploying a new IoT device or network, new and more vigorous demands will be placed on the networks. Applications and services such as high-speed wireless networks, high-definition IP video services, and real-time measuring systems require high-bandwidth connectivity. In addition, extremely low-latency applications, such as high-speed motion controls, demand high-speed connections to be indispensable. Figure 2 shows the architecture of Fog computing environment for the proposed spine-leaf datacenter integration.

The architecture shows the hierarchical arrangement of Fog edge devices throughout the network between sensors and the cloud core shown in Figure 1. In the architecture, IoT sensors are located at layer 1 of the Fog architecture stack. In reality, this is distributed in multitenanted geographical locations. In essence, the sensing environment propagates the generated data stream values to the Fog middleware using

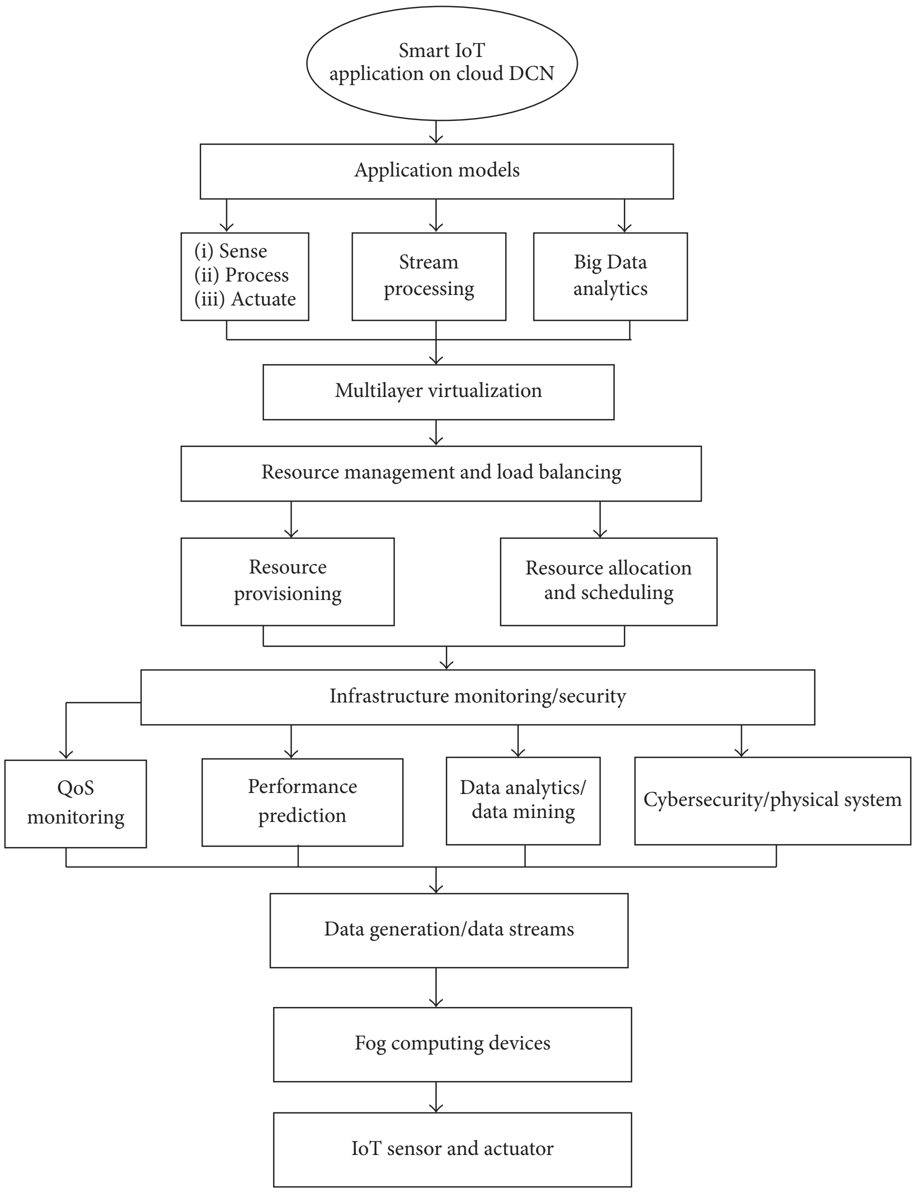
Smart IoT application on cloud DCN
Application models
(i) Sense
(ii) Process
(iii) Actuate
Stream processing
Big Data analytics
Multilayer virtualization
Resource management and load balancing
Resource provisioning
Resource allocation and scheduling
Infrastructure monitoring/security
QoS monitoring
Performance prediction
Data analytics/ data mining
Cybersecurity/physical system
Data generation/data streams
Fog computing devices
IoT sensor and actuator

Figure 2: Fog computing IoT architecture for spine-leaf datacenter integration.

intelligent gateways. This is used for extended distributed processing and filtering.

In layer 1, IoT actuators serve as a control system designed to communicate real-time deviations or changes in environments when the sensors detect any event. In this regard, the IoT data streams from sensors form the datasets for analytics. The Fog devices in Figure 2 basically host the application modules that are connected to sensors. An integrated service gateway links these sensors to the Internet. Besides, cloud resources provisioned on demand from geographically distributed datacenters are encapsulated in Figure 2.

The multilayer virtualization gives support for resource management and load balancing on the cloud. In this case, scheduling and resource provisioning are included for Fog and IoT environments.

At the infrastructure monitoring (components) layer, the QoS monitoring checks and ensures stable resource utilization, throughput, and availability of sensors and actuators. Also, it ensures that Fog devices and network elements are at an optimal level. Performance monitoring of the applications and services on the infrastructure is addressed by these components. Without resource management, the QoS expectations will be truncated while allowing for resource wastage. By providing active load balancing and scheduling, the available resources for workloads will be uniformly shared. Other profiles monitored are power and application program interfaces (APIs) for the IoT system. Finally, the architectural models supported for IoT applications are as follows:

(i) *Smart Sensing, Processing, and Actuation (SSPA).* In this case, the information gathered by sensors is emitted as data streams, which are acted upon by applications running on Fog devices. Hence, the resultant control signals are sent to actuators.

(ii) *Smart Stream Processing (SSP).* With the SP model, this uses its network of application modules running on Fog devices to continuously process data streams emitted from sensors. The generated information mined from the incoming streams is stored in datacenters for large-scale and long-term Big Data analytics.

The above considerations engendered the need for a flexible and scalable network infrastructure. This type of network can easily deploy applications from the cloud down to the Fog edge while serving myriads of devices joining the network. Furthermore, the IoT network infrastructure must be secured and scalable. Hence, resilience at scale, integrated security, and converged networking are the key IoT requirements for conceptual Fog conceptual framework in Figure 1. The benefits include scalable network for real-time analytics, support for cloud to Fog and edge data processing, and reliable QoS provisioning.

Considering the Fog architecture, some useful IoT network technologies that can be considered in the implementation stage from Cisco systems [33] include embedded service routers (Cisco 59xx series), switches Cisco 2020, Industrial Ethernet 4000 Series Switches, Cisco ASR 900 (i.e., Aggregation Services Router), and Cisco 1000 Series Connected Grid Router. These devices meet the IoT needs of the various market segments such as energy, manufacturing, oil and gas, utilities, transportation, mining, and public sector.

4. Spine-Leaf Network Topology

4.1. System Description. So far, the importance of allowing intrinsic data processing locally in a scalable network has been discussed. This is very important in supporting applications like smart traffic systems, energy grids, smart cars, health care systems, and so forth, as shown in Figure 1. With localized data processing/data offloading, it is possible to create an efficient network model that is relatively cost-effective. The main task of Fog offloading is to position information near the user at the network edge as shown in Figure 1. Figure 3 shows the proposed scalable SL-FCN. With such system, important alerts and other details would still be sent to the cloud.

Also, the use of spine-leaf Fog computing would relieve larger workloads, reduce latency issues, and make for smoother responses as more people engage the IoT layers. However, SL-FCN is proposed for scalability considering the Fog IoT era. This type of datacenter network is well positioned for data analytics and distributed data collection points. End services like setup boxes and access points can be easily hosted using SL-FCN. This improves QoS metrics generally. Moreover, since there is an increased focus on real-time massive data transfers as well as instantaneous data trajectory in the network, the legacy three-tier design within a datacenter can be replaced by the Fog leaf-spine design. This model is obviously adaptable to the continuously changing needs of services and applications in the Big Data domain, hence bringing about enterprise scalability of instantaneous stream workloads. The advantages are enumerated in Section 4.2.

4.2. Advantages of Spine-Leaf FCN. With Smart Sensing, Processing, and Actuation (SSPA) and Smart Stream Processing (SSP) in Figure 2, the highlighted advantages of the massively scaled Fog computing network are enumerated as follows:

(i) Massive scalability arising from multihypervisor virtualized systems with bandwidth optimized paths

(ii) Reduction in data movement across the network resulting in reduced congestion, cost, and latency

(iii) Elimination of disturbances resulting from centralized computing systems

(iv) Improved security of encrypted data from edge devices as the edge layer stays closer to the end user, reducing exposure to unfavourable elements

(v) Eliminating the core computing environment, thereby reducing centralized computing and point of failure instances

(vi) Enhance edge security, as data are encoded when moving towards the network edge

(vii) Integrated support for edge computing while providing low-latency response to end users

(viii) Using dynamic layer 3 routing for interconnecting various layers

(ix) Providing high levels of reliability and fault tolerance

(x) Consequent lower bandwidth consumption generally

(xi) Removal of STP between the legacy access and aggregation layers using dynamic layer 3 routing results in a much more stable environment

4.3. Spine-Leaf FCN Limitations. There are few issues regarding the proposed system. The only limitations of the proposed network are as follows:

(i) It literally introduces issues on the selection of cabling technology platforms for link paths.

(ii) The other major disadvantage arises from the use of layer 3 routing. This has obviously eliminated the spanning of VLANs (virtual LAN) across a network. In context, the VLANs in a spine-leaf FCN are localized to each individual leaf switch; as such, any VLAN microsegments left on a leaf switch will be inaccessible by the other leaf instances. Problems can arise from this type of scenario, for instance, when guest virtual machine mobility is introduced within a datacenter.

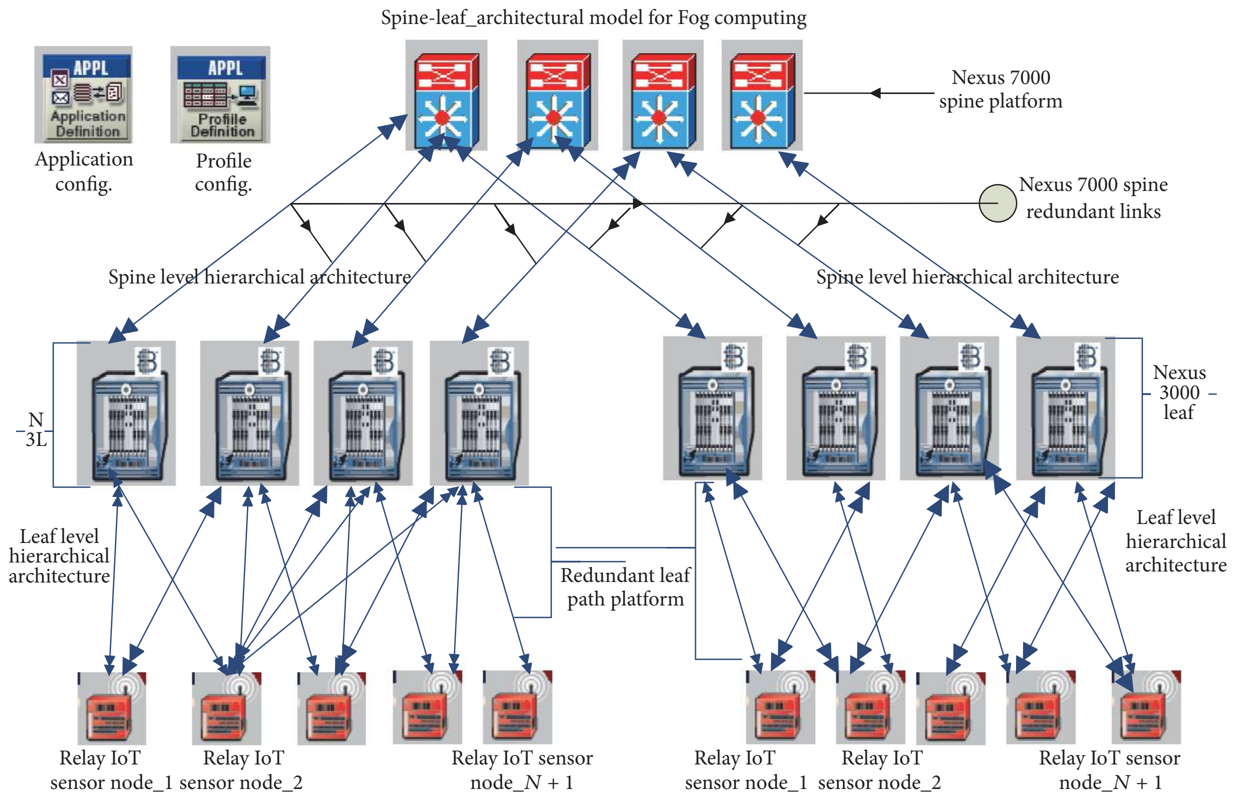
Spine-leaf_architectural model for Fog computing
APPL
Application Definition
Application config.
APPL
Profile Definition
Profile config.
Nexus 7000 spine platform
Nexus 7000 spine redundant links
Spine level hierarchical architecture
Spine level hierarchical architecture
N
3L
Nexus 3000 leaf
Leaf level hierarchical architecture
Leaf level hierarchical architecture
Redundant leaf path platform
Relay IoT sensor node_1
Relay IoT sensor node_2
Relay IoT sensor node_N + 1
Relay IoT sensor node_1
Relay IoT sensor node_2
Relay IoT sensor node_N + 1

Figure 3: Design of Fog computing datacenter using Cisco Nexus Platforms.

5. Design Implementation

The implementation of SL-FCN is constituted by the simulated Fog cloud entities and services. First, a description on how the elements of architecture are modelled is presented. In this paper, SL-FCN was developed using the event simulation functionalities found in CloudSim [34] and Riverbed Modeller Version 15.6.11 [35].

In this regard, entities in object palette CloudSim, like Nexus datacenter multilayer switches, communicate between each other by message passing operations (sending events) via the link paths as demonstrated in Figure 3. With the design setup, the super spine core layer is responsible for handling events between Fog computing components in CloudSim C++ class of Riverbed Modeller. The main classes of network topology as introduced in Figure 3 include Fog device sensors, physical topology, actuators, object placement mapping, application edge, controllers for Fog devices, and configuration application engine (Table 1).

5.1. Integration Techniques. Using the C++ library of the modeller simulator, the network application module placement strategies include the traditional cloud placement and Fog placement in the spine-leaf scenarios. The cloud placement strategy is based on the traditional cloud implementation where all modules of an application run in legacy three-tier datacenter. With the proper sensor placement, the sense-process-actuate applications are implemented by having the IoT sensors transmit sensed data to the cloud for real-time processing. After the processing, any other legitimate action can be enforced by the actuator.

Also, in the Fog layer placement scenario, application deployment is made close to the network edge as depicted in Figure 3. This is because the Fog devices such as smart gateways and access points placed close to the edge of the network lack robust computational stability to host bandwidth intensive applications.

Essentially, the redundant Nexus platform at the Fog layer is cascaded as much as possible to support Fog application modules placed between the network edge and the cloud. This is necessary to stabilize the QoS.

5.2. Spine-Leaf-FCN Graphical Interface. The SP-FCN network topology illustrated in Figure 3 has its graphic user interface built over the C++ library of the modeller simulator application logic. The interface allows for representing the actual physical elements such as Fog Nexus devices, sensors, actuators, and connecting links. The defining attributes of these entities are captured into the topology using the GUI application engine templates. After developing the design, this was saved and reloaded for testing from its trace file (JSON) format. With rich JAVA APIs, it is also possible to develop the same physical topologies. The SP-FCN has layer 1 sensors, smart gateways, and cloud virtual machines, as well as the link path connection depicted in Figure 3. For testing the scalable Fog IoT design, the basic steps for the simulation

Table 1: Description of the physical topology for IoT FCN.

Fog Nexus Router (spine node)	Cisco 7000
Fog Nexus Router (leaf node)	Cisco 3000
Total ISR/switch in topology	4 spine + 8 leaf (small); 4 spine + 16 leaf (medium); 8 spine + 16 leaf (small)
Fog devices	11
Wi-Fi access points	8
Fog C++ library	Enabled
Path link	40 GB
Multilayer virtualization	Type 1
Routing type	Dynamic layer-3
Redundancy	Active ($N + 1$)

are discussed below. The design was then analyzed to show the performance of the system.

(1) Using the imported C++ libraries, the physical entities were created while outlining their capabilities and specification configurations. The actual entities used include IoT RF sensors, Nexus gateways (ISR), cloud virtual machines, and link paths (40 GB links). These are connected layer by layer.

(2) The workload of the system was carefully realized by setting transmit rates of layer 1 sensor using transmit distribution attribute in SensorClass (sensor MAC).

(3) The definition of placement and scheduling policies that map application modules to Fog devices is made. In all cases, these policies in context only considered end-to-end processing latency, network availability, and network resource usage constraints. The traces file Module Placement and Controller (MPC) classes were used in implementing the required placement logic.

(4) The identification of QoS profile for workload is made so as to ascertain the network latency, availability, and network resource usage at incremental loading.

(5) The result analysis was carried out for the QoS metrics in an Excel worksheet after the design completion.

6. Performance Evaluation

This work will now present the two simulation case studies for scalable IoT network. In this regard, the simulation study was carried out for a period of 1000 seconds. The study focused on introducing a latency-sensitive application on the SL-FCN. The latency-critical application involves augmented event based sensing and data-offload interaction. In the design, the real-time processing in context requires that the application be hosted very close to the source of data for possible evaluations. This was done to show the impact of scalable SL-FCN for the event based application case study. In this regard, the efficiencies of two placement strategies (i.e., legacy cloud and Fog layer) were evaluated in terms of latency, network usage, and resource availability.

6.1. Evaluation of Case Study 1: Latency Profile for Scalable Spine-Leaf-FCN Placement. In analyzing latency metric, the simulation experiment for this case study was carried out for a period of 2 hours in extended Riverbed DES while taking cognizance of the report metrics collected. The results of the simulation experiment demonstrate how the different input workloads and placement strategies impact the overall end-to-end latency. In essence, each IoT edge device establishes a communication link while gaining access to the Internet through Wi-Fi gateways. This is then connected to the smart ISP gateway. For the purpose of testing SL-FCN performance, the topology sizes and the number of Wi-Fi gateways were varied while keeping the edge devices connected to each Wi-Fi gateway and ISR constant. The two configurations of physical topology simulated are the Fog placement and cloud placement in configuration model, namely, Scenario 1 and Scenario 2. Initially, each case had 8 Wi-Fi gateways and 10 Nexus ISR in Figure 1. For testing/validation in Figure 3, four identical Nexus 7000 devices were used as the spine platform for hosting the cloud placement, while eight Nexus 3000 Fog devices were used for the leaf platform which connects all the Fog and edge devices/applications. In this case, each gateway is connected to edge devices/applications for event based services. The performance of an application on the Fog depends on latencies of the links connecting the Fog devices. Hence, 40 GB Ethernet link was introduced in the simulation topology. A real-time communication between the edge devices and cloud domain hosting its services with efficient processing is very expedient in the event based communication/sensing. Any lag in real-time reporting will severely affect user experience at large. Figure 4 illustrates the average delay/latency in execution of data-offloading sequence. It was observed that latency execution dramatically increases for cloud application (about 87.5) and services placement. However, it decreases for edge-ward placement strategy (about 12.5%) where Fog devices are utilized for processing. This reduction is more significant when Fog topology size increases.

6.2. Evaluation of Case Study 2: Network Usage Profile for Scalable Spine-Leaf-FCN Placement. Figure 5 shows the network usage of the spine-leaf FCN. An increase in the number of devices connected to the application significantly increases the load on the network where both cloud and Fog

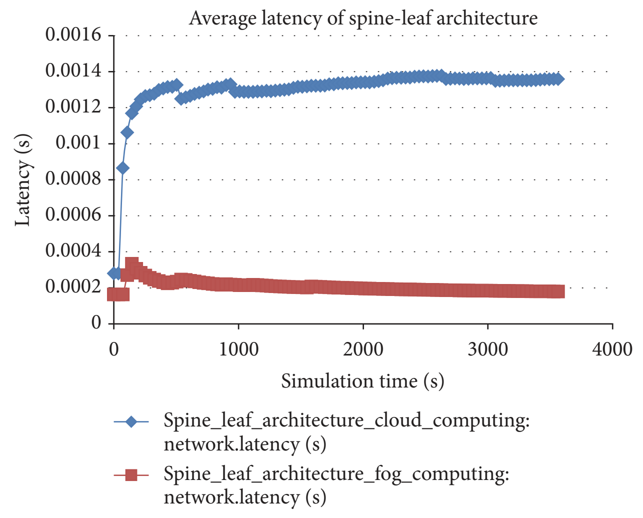

FIGURE 4: Average latency profile of SLFCN.

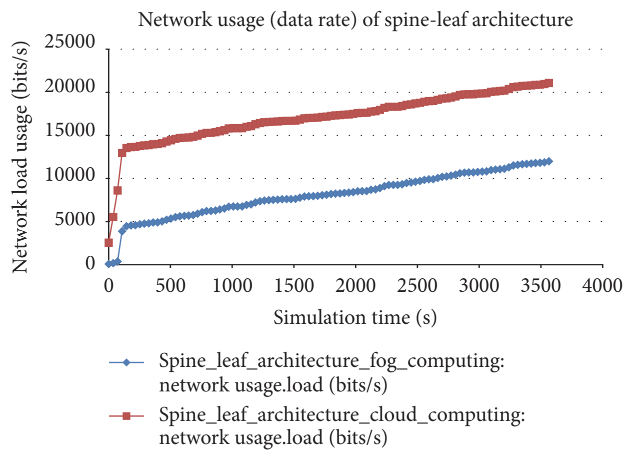

FIGURE 5: Average network usage profile of SLFCN.

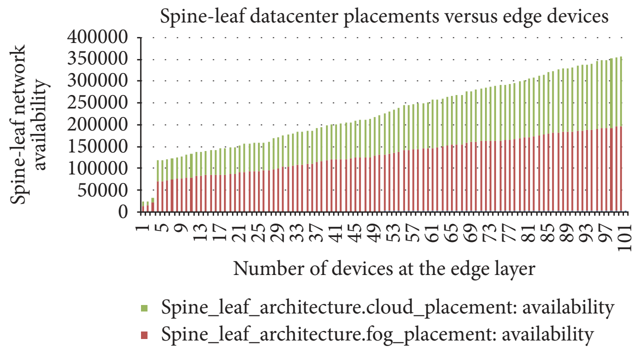

FIGURE 6: Spine-leaf DCN placement for services availability.

resources are used. As shown in Figure 5, when Fog devices are considered, the network usage considerably decreased. This is not the case for cloud layer placement or usage incidence. The network density seems to be very high relative to Fog computing. For network usage, SP-FCN offers self-learning, self-organization, and self-healing for massively scaled IoE scenario. This makes congestion and latency issues less significant while providing the necessary platform managing heterogeneous and distributed real-time services with minimal impact on QoS.

6.3. Evaluation of Case Study 3: Network Availability Profile for Scalable Spine-Leaf-FCN Placement. As shown in Figure 6, it was observed that as the number of devices connected to the network increases, the load on the network increases significantly in the case of cloud deployment. This is in contrast to Fog layer deployment. However, with the application placement strategies, the implication is that, with spine-leaf FCN, there will be high guarantee of network availability. Scalable network environments must not necessarily have high throughput; rather, stability remains a desirable consideration. Consequently, with spine-leaf FCN, scalability guarantee as well as optimal bandwidth utilization will obviously improve the network performance. This will invariably make multitenanting an interesting dimension in the IoT/IoE computing era.

7. Conclusion and Future Works

This paper has dealt with Fog computing in the context of scalable IoT datacenter network perform. This network is shown to complement cloud computing functionalities in meeting the demands of the emerging IoT stream processing paradigms. The spine-leaf FCN is a scalable model developed which works side by side with cloud computing datacenter networks. This can be used for data offloading and for high-end batch processing workloads in today's IoT era. Existing cloud networks have issues with policy framework for QoS management. Also, for the traditional datacenter tree-like structures, issues such as limited server-to-server connectivity, vulnerability to single point of failure, lack of agility, insufficient scalability, and smart resource fragmentation are still affecting the cloud domain. Hence, a conceptual IoT framework is developed. This was based on comprehensive architectural decomposition such as application models, multilayer virtualization, resource management/load balancing, infrastructural monitoring, data streams, and IoT edge sensors/actuators. To further validate the work, network usage, latency, and availability metrics were considered for both Fog and cloud spine-leaf architectures. The results show that scalable FCN can offer reliable QoS while remaining fault-tolerant for traffic workloads. It can be concluded that Fog computing and cloud computing will complement each other regardless of their merits and limitations. Even if Fog computing grows exponentially in the emerging network contexts, scalability and low-latency processing are key issues that must be taken into consideration for satisfying high-end computing demands. This will invariably reduce cost.

Future work will focus on using a production spine-leaf network environment for validating the IoT data stream processing. Other areas of future research include priority-aware resource management strategies for multitenant environments, modelling and analysis of failure modes in Fog and edge devices considering scheduling, and recovery policies for a wide range of applications. Also, work on the validation Fog cloud virtualization techniques (Full, Para, and OS virtualization) in IoT environments is necessary.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this article.

Acknowledgments

This research was carried out as an extended work on Distributed Cloud Computing Network for SGEMS/EETACP project commissioned by the Department of Electronic Engineering, University of Nigeria Nsukka.

References

[1] H. Gupta, A. V. Dastjerdi, S. K. Ghoshy, and R. Buyya, "iFogSim: a toolkit for modeling and simulation of resource management techniques in internet of things," *Edge and Fog Computing Environments*, pp. 1–22, 2016.

[2] S. Kuppusamya, V. Kaniappanb, D. Thirupathic, and T. Ramasubramanian, "Switch Bandwidth Congestion Prediction in Cloud Environment," pp. 235–243, 2015, Proceedings of the 2nd International Symposium on Big Data and Cloud Computing (ISBCC '15), Published by Elsevier B.V. 2015.

[3] Y. Zhang and N. Ansari, "On architecture design, congestion notification, TCP incast and power consumption in data centers," *IEEE Communications Surveys and Tutorials*, vol. 15, no. 1, pp. 39–64, 2013.

[4] K. C. Okafor, *development of a model for smart green energy management system using distributed cloud computing network [Ph.D. thesis]*, Department of Electronic Engineering, University of Nigeria Nsukka, 2015.

[5] M. A. Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *Proceedings of the SIGCOMM*, Seattle, Wash, USA, 2008.

[6] A. Greenberg, N. Jain, S. Kandula et al., "Vl2: a scalable and flexible data center network," in *Proceedings of the SIGCOMM*, vol. 12, Barcelona, Spain, 2009.

[7] R. Mysore, A. Pamboris, and N. Farrington, "Portland: a scalable fault-tolerant layer 2 data center network fabric," in *Proceedings of the SIGCOMM*, Barcelona, Spain, 2009.

[8] J. Kim, W. J. Dally, S. Scott, and D. Abts, "Technology-driven, highly-scalable dragonfly topology," in *Proceedings of the 35th International Symposium on Computer Architecture (ISCA '08)*, pp. 77–88, Beijing, China, June 2008.

[9] B. Arimilli, R. Arimilli, V. Chung et al., "The PERCS high-performance interconnect," in *Proceedings of the 18th IEEE Symposium on High Performance Interconnects (HOTI '10)*, pp. 75–82, Mountain View, Calif, USA, August 2010.

[10] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, "Dcell: a scalable and fault-tolerant network structure for data centers," in *Proceedings of the SIGCOMM*, Seattle, Wash, USA, 2008.

[11] D. Li, C. Guo, H. Wu, K. Tan, Y. Zhang, and S. Lu, "FiConn: using backup port for server interconnection in data centers," in *Proceedings of the 28th Conference on Computer Communications (IEEE INFOCOM '09)*, pp. 2276–2285, Rio de Janeiro, Brazil, April 2009.

[12] C. Guo, G. Lu, D. Li et al., "Bcube: a high performance, server-centric network architecture for modular data centers," in *Proceedings of the SIGCOMM*, Barcelona, Spain, 2009.

[13] D. Guo, T. Chen, D. Li, Y. Liu, X. Liu, and G. Chen, "BCN: expansible network structures for data centers using hierarchical compound graphs," in *Proceedings of the 30th IEEE International Conference on Computer Communications (INFOCOM '11)*, pp. 61–65, Shanghai, China, April 2011.

[14] J. Hamilton, "An architecture for modular data centers," in *Proceedings of the 3rd CIDR Conference*, pp. 306–313, Pacific Grove, Calif, USA.

[15] M. M. Waldrop, "Data center in a box," *Scientific American*, vol. 297, no. 2, pp. 90–93, 2007.

[16] D. Li, M. Xu, H. Zhao, and X. Fu, "Building mega data center from heterogeneous containers," in *Proceedings of the 19th IEEE International Conference on Network Protocols (ICNP '11)*, pp. 256–265, Vancouver, Canada, October 2011.

[17] H. Wu, G. Lu, D. Li, C. Guo, and Y. Zhang, "MDCube: a high performance network structure for modular data center interconnection," in *Proceedings of the ACM Conference on Emerging Networking Experiments and Technologies (CoNEXT '09)*, pp. 25–36, Rome, Italy, December 2009.

[18] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *Proceedings of the IEEE INFOCOM*, San Diego, Calif, USA, March 2010.

[19] P. M. Wells, K. Chakraborty, and G. S. Sohi, "Dynamic heterogeneity and the need for multicore virtualization," *ACM SIGOPS Operating Systems Review*, vol. 43, no. 2, pp. 5–14, 2009.

[20] P. M. Wells, K. Chakraborty, and G. S. Sohi, "Mixed-mode multicore reliability," in *Proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '09)*, pp. 169–180, March 2009.

[21] Lin Wang, Fa Zhang, Athanasios V. Vasilakos, and C. H. Z. Liu, *Joint Virtual Machine Assignment and Traffic Engineering for Green Data Center Networks*, Greenmetrics, Pittsburgh, Pa, USA, 2013.

[22] G. Wang and T. S. E. Ng, "The impact of virtualization on network performance of Amazon EC2 Data Center," in *Proceedings of the IEEE INFOCOM*, March 2010.

[23] Amazon ec2, http://aws.amazon.com/ec2/.

[24] Gogrid, http://www.gogrid.com/.

[25] P. Barham, B. Dragovic, K. Fraser et al., "Xen and the art of virtualization," in *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP '03)*, October 2003.

[26] X. Huang and Y. Peng, "A Novel Method of Fault-Tolerant Decentralized Lookup Service for the Cloud Computing," vol. 29, pp. 3234–3239, 2012, Proceedings of the International Workshop on Information and Electronics Engineering (IWIEE '12).

[27] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, "Dryad: distributed data-parallel programs from sequential building blocks," *ACM SIGOPS Operating Systems Review*, vol. 41, no. 3, 2007.

[28] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: cluster computing with working sets," *HotCloud*, 2010.

[29] S. Bykov, A. Geller, G. Kliot, J. R. Larus, R. Pandya, and J. Thelin, "Orleans: cloud computing for everyone," in *Proceedings of the 2nd ACM Symposium on Cloud Computing (SOCC '11)*, p. 16, ACM, October 2011.

[30] J. Ekanayake, H. Li, B. Zhang et al., "Twister: a runtime for iterative MapReduce," in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing (HPDC '10)*, pp. 810–818, June 2010.

[31] P. Bhatotia, A. Wieder, R. Rodrigues, U. A. Acar, and R. Pasquini, "Incoop: MapReduce for incremental computations," in *Proceedings of the 2nd ACM Symposium on Cloud Computing (SOCC '11)*, ACM, October 2011.

[32] A. Shaout and S. Sreedharan, "fault-tolerant storage system for cloud datacenters," in *Proceedings of the 13th International Arab Conference on IT (ACIT '12)*, pp. 241–248, December 2012.

[33] Cisco IoT System, http://www.cisco.com/go/iotsystem, 2015.

[34] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software—Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011.

[35] Riverbed Modeler Academic Edition 17.5 PL6: Available Online: https://splash.riverbed.com/community/product-lines/steelcentral/university-support-center/blog/2014/06/11/riverbed-modeler-academic-edition-release.