Context-Aware Fog Computing Implementation for Industrial Internet of Things

Adel Atieh
Faculty of Engineering and IT
University of Technology, Sydney (UTS)
Sydney, Australia
adel.atieh@student.uts.edu.au

Priyadarsi Nanda
Faculty of Engineering and IT
University of Technology, Sydney (UTS)
Sydney, Australia
priyadarsi.nanda@uts.edu.au

Manoranjan Mohanty
Faculty of Science
University of Technology, Sydney (UTS)
Sydney, Australia
manoranjan.mohanty@uts.edu.au

***Abstract*—The connectivity of devices has increased in the last decade enabling multiple innovative applications and solutions to serve industries and societies. This has solved multiple challenges and facilitated the improvement of methodologies and techniques adapted by humanity. One of the newly created paradigms that changed industries and technology is the Industrial Internet of Things (IIoT). IIoT is currently being adapted by various industries creating interactive supply chain ecosystems through the use of cloud computing. The size and distributions of these ecosystems introduced latency and Quality of Service (QoS) issues for edge devices sending data to the cloud. This research paper explores a paradigm called "Fog Computing" which aims to reduce the latency between IIoT devices and the cloud by deploying a "cloud-like" computing layer closer to the IIoT devices. In addition, a Context-Aware implementation of fog computing is proposed in this paper to provide the most optimised service to edge devices. Furthermore, this paper includes various experiments that examine the different context-awareness perspectives this paper proposes for fog computing. The results and outcomes of these experiments show reduction in latency and automated resource scaling from the use of context-awareness with fog computing over cloud computing for IIoT.**

***Index Terms*—Fog Computing, Industrial Internet of Things (IIOT), Context-Aware, Cloud Computing, Quality of Service (QoS)**

I. Introduction

Internet of Things (IoT) is a digital model that explores the enablement of connectivity between objects and devices through the Internet. These devices include home consumer devices, commercial devices and industrial devices. By connecting large number of devices together and collecting enormous data from every device, businesses and organisations become more aware of their environments and ecosystems. This enables the improvement of existing products and delivers smarter solutions to their consumers and partners [1].

The large amount of data collected has assisted organisations in realising their true potential and influenced industries through operation enhancement and integrations for more customer satisfaction. Due to this, the IIoT model started to be utilized widely across organizations to enhance monitoring and connectivity between organization assets [2]. This involves the replacement of currently deployed Industrial Control Systems (ICS) environments with smarter and more interactive devices to communicate with next-generation IT systems. ICS environments are deployed in critical infrastructures serving multiple sectors. These systems require real-time interactions between devices in order to function without any issues. Any errors occurring in these environments can have significant physical consequences and impact on national safety and services provided to the nation as a whole [3], [4]. Therefore, ensuring appropriate network and communication is essential to maintain sufficient functionality of these systems.

This research paper proposes a fog computing solution powered by Context Awareness API (CAAPI) that would ensure IIoT devices are connected to the closest fog servers. Moreover, to establish the cloud capabilities within Fog Computing, Kubernetes is used to run and autoscale containerised applications. Such implementation ensures appropriate resources are present to be utilised by the solution.

The remaining of this paper is organized into six sections. Section II presents existing works on fog computing and conducts a thorough review of these works. Section III explains and discusses the existing issues with IoT environments connected to cloud and goes through the proposed fog computing design and its context awareness capabilities. Section IV describes the experimental setup and implementation of the IIoT fog computing solution Proof of Concepts (PoCs). Section V shows the results produced by the PoCs and Section VI provides an analysis of these results. Finally, Section VII draws conclusions from the results on the use of fog computing and discusses future work and capabilities that can be added to the Fog Computing solution.

II. Literature Review

A. Cloud Computing

Cloud computing is a service offering that has been introduced by technology providers such as Amazon Web Services (AWS), Microsoft and IBM to provide computing, storage and network resources to organizations [5]. Along with the advances in technology and the introduction of IoT to various industries, communication technologies are also evolving to support the next generation of smart ecosystems. However, cloud computing has inherent physical limitations related to the long propagation distance between the end device and the cloud data centres. Also, the high number of routing hops to the cloud data centres can result in multiple places of processing delays adding to the latency related to the physical distance [6].

B. Fog Computing

Fog computing is defined as a virtual layer providing services for end devices and acts as the broker layer between the end devices and the cloud [7]. It is a technology paradigm that aims to extend the capabilities of the cloud to the IoT devices [8]. It provides programmability and orchestrated management and automation of services to the IIoT solution. Multiple fog computing use cases have been mentioned in various papers covering Smart Grids [9], Connected vehicles [9] and Healthcare [10]. Fog computing aims to provide various capabilities to the IoT deployments that focus on the following aspects [11], [12]:

- Distributed applications
- Low-latency and location-based awareness
- Context-Awareness
- Scalable according to the resource utilization

As the fog computing layer is made close to the end devices, low latency is achieved with greater aspect of security and network performance. This also reduces the total bandwidth cost of networks leading to a more "greener" technological ecosystem [7], [13]. Services provided by the cloud including computation, network and storage services introduce performance overheads and bottlenecks as they are being delivered centrally [8].

C. Context-Awareness

Context-awareness is a characteristic that is implemented through the collection of data from various sources and thereafter analysed and modelled to gain different perspectives of the deployed devices [14]. There are various definitions for context-awareness, however the following definition is relevant to this research [15]:

"A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task."

As mentioned in [14], there are two different types of context that define the level of perspective and depth of information gained from the data collected and are as follows:

- Primary context: Any information extracted from the collected data without performing any data analysis and/or operations on the data (Eg. GPS Coordinates, CPU utilization, etc).
- Secondary context: Any information retrieved as a result of analyzing and computing primary context data. This includes the distance between two different IIoT devices using their GPS coordinates.

One of the common methods used to standardise interactions between different devices and applications is the use of REST (REpresentational State Transfer) Application programming interface (API). The use of REST API allows interactivity between different systems using HTTP request methods. An example of this interactivity is the use of REST API calls to interact with Fog Computing gateways [16].

III. Current State of Research & Research Questions

As it currently stands, IoT ecosystems connect to cloud to send large amounts of data where computation and data storage is performed. However, latency sensitive applications suffer from factors that affect the Quality of Service (QoS), Quality of Communications (QoC) and saturations such as environmental factors [17]. These issues can affect IIoT systems that provide critical services that require real-time communication and sustainable management throughout their life-cycle [18].

Previous studies and research papers have discussed and explored Fog computing and demonstrated its potential over cloud computing. Various latency and data rate evaluations were performed by [16] using fog computing and cloud computing lab setups. Results of these evaluations have revealed the performance improvement using fog computing over cloud computing due to the distance difference. Furthermore, the solution proposed in [19] utilises the close proximity of fog computing to offload mobile device computation and storage of data to reduce the load on mobile devices. There have been multiple models and frameworks that were explored and implemented by different researchers for fog computing. T.Gia [10] proposed a fog computing solution for a real-life healthcare use-case. Moreover, M.Aazam [20] proposed a fog computing model that focuses on dynamic resource estimation, reservation and pricing.

While these features explored through the above papers enable solutions to improve and become more efficient, context-awareness is required to combine all of these features and utilise their metrics. This will allow fog computing to reach its potential and thus benefit its hosted solutions. Our paper aims to examine the following research questions:

- How can context-awareness help improve the performance of a fog computing solution over a cloud computing solution?
- What are the key aspects that can be considered for context awareness in fog computing?
- What are the schemes that can allow fog computing to establish context awareness of the environment?

IV. Proposed Solution Design

In this paper, we propose a fog computing solution that targets the challenges presented by cloud computing by deploying context-aware computing clusters close to IIoT devices. In addition, the proposed fog computing solution is designed and implemented similar to cloud computing by utilizing autoscaling and load-balancing capabilities. The proposed solution serving IIoT devices consists of the following three technology layers similar to this paper [12]:

- Edge Layer: This layer involves all the different types of industrial field controllers such as PLCs, RTU and all other sensors that generate data or receive digital binary commands to perform actions.
- Fog Layer: This layer provides computing and storage services to the edge layer with inter-connectivity through

various networking capabilities. Upon receiving data from the Edge layer, the Fog layer will perform analysis on the data, filtering and cleaning of data.
- Cloud Layer: This layer hosts CAAPI that controls the edge layer and fog computing systems centrally through the continuous monitoring of systems.

In order to provide the key services running on fog computing to the edge layer efficiently using context-awareness, the following components are used:

- Fog Gateway
- Microservices clusters
- Context Awareness API (CAAPI)

The fog layer is distributed across different locations consisting of Kubernetes clusters that aim to maintain the availability of the applications. Figure 1 shows the microservices design of the fog and cloud layers.

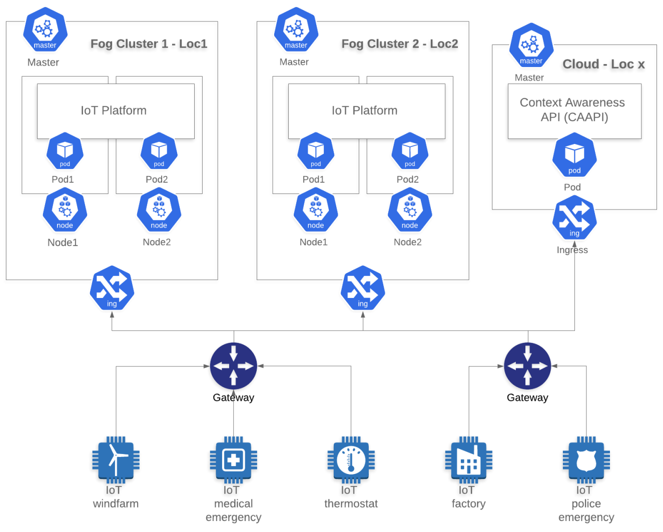

Fig. 1. Fog & Cloud Microservices Solution Design

A. Fog Gateway

Fog gateways are determined to be one of the key components of the fog computing architecture. Fog gateways are positioned close to the end devices to provide them access to computing resources. CAAPI is used to interact with the fog gateways through API calls and create routes for specific IIoT devices to their closest fog cluster.

B. Microservices Clusters

Microservices clusters are responsible for running, maintaining and scaling up IIoT core applications automatically using Kubernetes. Kubernetes developed by Google is used for simulating fog computing, taking advantage of the various orchestration and load scheduling capabilities to provide highly scalable fog computing implementation with automated geographic awareness [19], [21].

C. CAAPI

CAAPI is a REST API server hosted on the cloud to interact with the edge layer to direct IIoT devices to the appropriate fog cluster based on location and resource metrics. Context-Awareness is achieved through the collection of the primary context parameters shown in Table 1 below.

These primary context parameters are combined to construct the secondary context parameters shown in Table 2. Using these secondary context parameters, the following awareness perspectives are established:

- Infrastructure Awareness
- Geographic Awareness
- Performance Awareness

1) Infrastructure Awareness: Microservices clusters consist of multiple nodes to ensure hardware redundancy. Each node hosts at least one pod running the required IIoT application(s). A Horizontal Pod Autoscaler (HPA) is deployed in Kubernetes to scale up the number of the IIoT application pods on based on the chosen monitored metrics. The scaling up is performed using the following algorithm [22]:

$$P_d = P_c * (\frac{M_c}{M_d})$$

- P_c : The current number of pods

TABLE I
PRIMARY CONTEXT PARAMETERS

Parameter	Source	Summary
GPS coordinates	IIoT & fog server	Longitude and latitude coordinates of an IIoT device and a fog server
CPU	Fog server	CPU usage percentage of the fog server
RAM	Fog server	RAM usage percentage of the fog server
Request time - sent	IIoT	The time at which a data publish is sent to a Fog Server
Request time - received	Fog server	The time at which a data received by a Fog Server
Requests per minute	Fog server	The number of requests a fog server receives from IIoT devices per minute

TABLE II
SECONDARY CONTEXT PARAMETERS

Parameter	Data	Summary
Distance	• GPS coordinates (fog server) • GPS coordinates (IIoT device)	Distance between an IIoT device and a fog server.
Latency	• Request time – received (fog server) • Request time – sent (IIoT)	Estimated time it takes a data request to reach a fog Server from an IIoT device and vice versa.

- M_c : The current value for the measured metric
- M_d : The desired value for the measured metric
- P_d : The required number of replicas to reach and maintain M_d

Autoscaling aims to emulate the PaaS & SaaS implementation of cloud service offering where these services autoscale depending on the demand and utilization of cloud resources.

2) Geographic Awareness: The GPS coordinates are sent to the CAAPI server from IIoT devices. These coordinates can be referred to as GPS coordinate A for an IIoT device and GPS coordinate B for a fog server cluster. In order to compute the fog server clusters' distances to these IIoT devices we used the Haversine formula [24], [25]:

$$a = \sin^2(\phi B - \frac{\phi A}{2}) + \cos(\phi A) * \cos(\phi B) * \sin^2(\lambda B - \frac{\lambda A}{2})$$

$$c = 2 * a * \tan 2(\sqrt{a}, \sqrt{1-a})$$

$$d = R * c$$

- ϕA : Latitude coordinate of point A (radians)
- ϕB : Latitude coordinate of point B (radians)
- λA : Longitude coordinate of point A (radians)
- λB : Longitude coordinate of point B (radians)
- a : Haversine of angle c
- c : c is the central angle in radians
- d : distance between GPS coordinates A and coordinates B in Kilometres (Km)

Once the minimum distance is found among the calculated distances, the fog server cluster with the minimum calculated distance is chosen to communicate with the IIoT device. By keeping track of the GPS coordinates of the IIoT devices and the fog server clusters, the fog layer establishes and gains awareness of the environment.

3) Performance Awareness: Latency is calculated by continuously recording and calculating the difference between the times of sending and receiving packets between IIoT devices and fog servers. An average is then calculated to represent an average latency per fog server. This represented using the following equation:

$$\bar{t} = \frac{\sum_{n=1}^{N} t_{Fog_n} - t_{IIoT_n}}{N}$$

- t_{IIoT} : The recorded time on IIoT device for sending the request
- t_{Fog} : The recorded time on fog server for receiving the request
- n : the number of the current transaction
- N : The total number of transactions
- $\bar{t}$: Calculated average latency per fog server

V. Experimental Design & Implementation

As this research explores fog computing and our context awareness features, it is important to compare them to the functionality and performance of cloud computing. The main objectives of this experimental design are as follows:

- Measure and compare the latency between cloud computing and fog computing.
- Examine the Infrastructure Awareness of fog computing.
- Examine the Geographic Awareness capability of fog computing.

In order to achieve these objectives, the experimental design for the proposed solution consisted of a fog lab setup and a cloud lab setup. The solution implementation examined the functionality of the experimental design and recorded the results observed from the following tests:

- Latency Test
- Geographic Awareness Test
- AutoScaling Test

A. Experimental Design

1) Fog Lab Setup: In our experimental set-up we use ThingsBoard as the IoT platform. We deployed Kubernetes for data analysis and collection platform for the IIoT sensors. This Kubernetes deployment makes use of 2 pods with ThingsBoard Docker containers deployed in 2 nodes that are managed by a master. The 2 nodes are Virtual Machines (VMs) with 1 vCPU and 2GB RAM each. These pods are exposed externally as a service using "NodePort" exposure. A Raspberry Pi 3 model B+ board is used as the IIoT client which runs a Python Script to communicate with the ThingsBoard servers. The network connectivity was established via 2.4GHz wireless LAN using Wi-Fi.

2) Cloud Lab Setup: Google Cloud was chosen for the cloud setup as it has a native Kubernetes engine very similar to the local Kubernetes engine. The setup is very similar to the fog virtual lab setup except the service exposure to the outside world is implemented using "LoadBalancer". This involves the use of a LoadBalancer for managing the traffic load between the 2 nodes. We have chosen *australia-southeast1-a* as the Google Cloud zone which is located in *Sydney, Australia* in order to get the optimum latency between our IIoT client and our cloud setup. The internet bandwidth used for connecting the IIoT client and the cloud setup was as follows:

- Download $\approx$ 50 Mbps
- Upload $\approx$ 14 Mbps

B. Implementation

1) Latency Test: In this test we compare the number of hops and the total latency for both deployments. As the fog computing layer is closer to the IoT layer, we expect a latency reduction with lower number of hops to the destination. The following network latency and measurements are performed:

- Traceroute: is used to compare the hop count between the fog and cloud deployments.
- Netperf: is used to send traffic with various packet sizes to the fog and cloud deployments and measure the Route Trip Time (RTT) to measure the latency.

2) Geographic Awareness: In this test we explore the implementation of the geographic awareness characteristic of CAAPI. This test is implemented using the fog virtual lab by hard-coding the GPS coordinates on the IIoT device and the Kubernetes clusters. A fog gateway was used to direct the IIoT client to the closest Kubernetes cluster. We chose VYOS for the fog gateway which is an open-source network gateway based on Debian Linux distribution. VYOS has an open API that allows the addition of routes and firewall rules [26]. We deployed CAAPI in the Cloud layer using Python running a Flask Server as a REST API server.

3) Infrastructure Awareness: This test examines the autoscaling behavior of CAAPI implemented by Kuberentes. HPA was deployed on a Kubernetes fog cluster to measure Thingsboard Application pods resource metrics.The key metric value that we chose to maintain by HPA is the following:

- CPU Utilisation Percentage $\approx 50\%$

VI. Experimental Results

A. Latency Test

1) Traceroute Test: The traceroute test for the cloud deployment showed that there were multiple ISP hops between the IIoT client and the cloud server. In contrast, the traceroute test for the Fog servers reached the Fog cluster through the fog gateway hop only. Table 3 shows the count of hops per lab setup.

TABLE III
Traceroute Test Results

Number of hops per lab setup	
Fog Lab Setup	Cloud Lab Setup
1	7

2) Netperf Test: The Netperf test was performed by sending continuous data streams to the fog and cloud lab clusters with various data packet sizes. As shown in Figure 2, the mean latency for the cloud deployment is nearly 4-5 times higher than the fog deployment mean latency. This is also reflected by the measured transaction rates for both environments shown in Figure 3. The transaction rate between the IIoT client and the Fog Computing deployment is around 3-4 times higher than the transaction rate for the Cloud deployment.

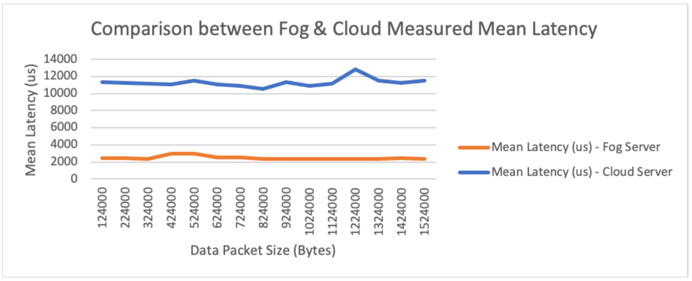

Fig. 2. Mean Latency Comparison between the local lab (Fog) and the cloud lab

B. Geographic Awareness

This test was performed in various methods to explore the various use cases of IIoT devices. When the IIoT device had

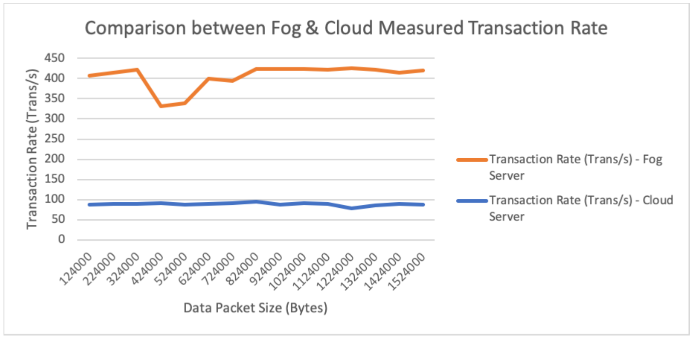

Fig. 3. Transaction Rates Comparison between the local lab (Fog) and the cloud lab

static GPS coordinates, CAAPI sent the closest fog server's IP address and port number to the IIoT client. Once the IIoT device GPS coordinates started varying, CAAPI was also able to redirect the IIoT client to the closest fog server based on the new GPS coordinates. However, the connectivity redirection delay depended on the hard-coded period for sending the IIoT client's GPS coordinates to the CAAPI server which was 5 seconds. This means that the CAAPI server had a 5 seconds delay until it becomes aware of the new GPS coordinates. This can be changed depending on the nature of the IIoT environment.

C. AutoScaling

The HPA engine was monitoring the utilization of CPU by the ThingsBoard application. A load generator was used for increasing the load to trigger the autoscaler to scale the number of pods. Kubernetes HPA was configured to have a minimum of 1 application pod and scale up to 10 applications pods. Figure 4 shows the observed Fog server autoscaling behaviour by the Kubernetes engine through monitoring the number of application pods replicas against the CPU usage percentage over time.

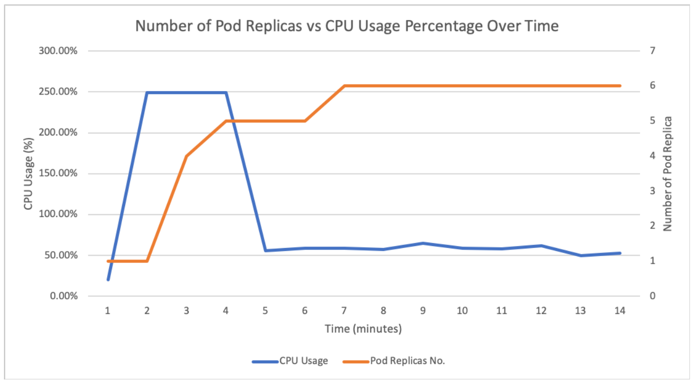

Fig. 4. Fog Computing Autoscaling behaviour by Kubernetes Engine over time

VII. Analysis

As expected from the latency test and similarly with the results from [19], the measured latency between the IIoT client and the Fog lab setup was around 4-5 times lower than the cloud lab setup due to the shorter distance. The major support factors that can allow fog computing to continuously succeed is the context-awareness manifested through the implementation of the different awareness perspectives explored in this paper. As observed during the Geographic and Infrastructure awareness tests, CAAPI was able to connect the IIoT client to the closest fog server and Kubernetes autoscaled the computing resources as required. Also, by continuously measuring the latency, CAAPI would be able to choose the most appropriate fog server for an IIoT device based on its own performance reputation. This is done by measuring average latency for a particular fog server across the various communications it has with all IIoT devices. This means that Performance Awareness can be combined with Geographic Awareness to improve the context-awareness decisions made by CAAPI. In addition, Infrastructure awareness can also be combined with Performance awareness to achieve a better QoS for all client devices. This leads to a Multi-dimensional context-awareness by combining these awareness perspectives to further improve the overall efficiency and performance of fog computing over cloud computing.

VIII. Conclusion

The convergence of technology and the internet is growing everyday leading to a continuous increase of data generation. IIoT environments rely on the convergence of IT and OT environment in order to send their sensors data and manage IIoT devices centrally. This requires low latency networks in order to accommodate for latency-sensitive environments. Fog computing enables this convergence with close physical proximity to the edge enhanced with context awareness to reduce the latency to an optimal rate. Further work needs to be done on designing a complete REST API based on the OpenAPI standard for CAAPI. More importantly, security needs to be an essential component in this design in order to authenticate, authorize and audit all the devices and their activities with CAAPI.

References

[1] Sisinni, E., Saifullah, A., Han, S., Jennehag, U. & Gidlund, M. 2018, 'Industrial Internet of Things: Challenges, Opportunities, and Directions', IEEE Transactions on Industrial Informatics, vol. 14, no. 11, pp. 4724-34.

[2] Libow, E., Indurkhya, G., Kreger, H., Hahn, T., Niblett, P., Mike Edwards, Wallace, T.S.S., Luthra, T., Menon, R., Schalk, K., Koupman, E., Daly, G., Flaherty, R., Noller, D. & Kiradjiev, P. 2016, The IBM Advantage for Implementing the CSCC Cloud Customer Reference Architecture for Internet of Things (IoT), IBM.

[3] Nazir, S., Patel, S. & Patel, D. 2017, 'Assessing and augmenting SCADA cyber security: A survey of techniques', Computers & Security, vol. 70, pp. 436-54.

[4] Hentea, M. 2008, 'Improving security for SCADA control systems', Interdisciplinary Journal of Information, Knowledge, and Management, vol. 3, no. 1, pp. 73-86.

[5] Duan, Q., Yan, Y. & Vasilakos, A.V. 2012, 'A Survey on Service-Oriented Network Virtualization Toward Convergence of Networking and Cloud Computing', IEEE Transactions on Network and Service Management, vol. 9, no. 4, pp. 373-92.

[6] Luan, T.H., Gao, L., Li, Z., Xiang, Y., Wei, G. & Sun, L. 2015, 'Fog computing: Focusing on mobile users at the edge', arXiv preprint arXiv:1502.01815.

[7] Aazam, M. & Huh, E. 2016, 'Fog Computing: The Cloud-IoTIoE Middleware Paradigm', IEEE Potentials, vol. 35, no. 3, pp. 40-4.

[8] Sarkar, S., Chatterjee, S. & Misra, S. 2018, 'Assessment of the Suitability of Fog Computing in the Context of Internet of Things', IEEE Transactions on Cloud Computing, vol. 6, no. 1, pp. 46-59.

[9] Bonomi, F., Milito, R., Zhu, J. & Addepalli, S. 2012, 'Fog computing and its role in the internet of things', paper presented to the Proceedings of the first edition of the MCC workshop on Mobile cloud computing, Helsinki, Finland, ¡https://doi.org/10.1145/2342509.2342513¿.

[10] Gia, T.N., Jiang, M., Rahmani, A., Westerlund, T., Liljeberg, P. & Tenhunen, H. 2015, 'Fog Computing in Healthcare Internet of Things: A Case Study on ECG Feature Extraction', 2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing, pp. 356-63.

[11] Iorga, M., Feldman, L., Barton, R., Martin, M.J., Goren, N.S. & Mahmoudi, C. 2018, Special Publication (NIST SP) - 500-325 Fog Computing Conceptual Model National Institute of Standards and Technology (NIST).

[12] Hu, Y.C., Patel, M., Sabella, D., Sprecher, N. & Young, V. 2015, 'Mobile edge computing—A key technology towards 5G', ETSI white paper, vol. 11, no. 11, pp. 1-16.

[13] Zhu, C., Leung, V.C.M., Shu, L. & Ngai, E.C. 2015, 'Green Internet of Things for Smart World', IEEE Access, vol. 3, pp. 2151-62.

[14] Perera, C., Zaslavsky, A., Christen, P. & Georgakopoulos, D. 2013, 'Context aware computing for the internet of things: A survey', IEEE communications surveys & tutorials, vol. 16, no. 1, pp. 414-54.

[15] Abowd, G.D., Dey, A.K., Brown, P.J., Davies, N., Smith, M. & Steggles, P. 1999, 'Towards a better understanding of context and context-awareness', International symposium on handheld and ubiquitous computing, Springer, pp. 304-7.

[16] Dastjerdi, A.V., Gupta, H., Calheiros, R.N., Ghosh, S.K. & Buyya, R. 2016, 'Fog computing: Principles, architectures, and applications', Internet of things, Elsevier, pp. 61-75.

[17] Wang, L. & Ranjan, R. 2015, 'Processing Distributed Internet of Things Data in Clouds', IEEE Cloud Computing, vol. 2, no. 1, pp. 76-80.

[18] Sajid, A., Abbas, H. & Saleem, K. 2016, 'Cloud-Assisted IoT-Based SCADA Systems Security: A Review of the State of the Art and Future Challenges', IEEE Access, vol. 4, pp. 1375-84.

[19] M. A. Hassan, M. Xiao, Q. Wei, and S. Chen, "Help your mobile applications with fog computing," in Proc. 12th Annu. IEEE Int. Conf. Sens. Commun. Netw. Workshops (SECON Workshops), Seattle, WA, USA, 2015, pp. 1–6.

[20] M. Aazam and E. Huh, "Fog Computing Micro Datacenter Based Dynamic Resource Estimation and Pricing Model for IoT," 2015 IEEE 29th International Conference on Advanced Information Networking and Applications, Gwangju, Korea (South), 2015, pp. 687-694, doi: 10.1109/AINA.2015.254.

[21] Santos, J., Wauters, T., Volckaert, B. & Turck, F.D. 2019b, 'Towards Network-Aware Resource Provisioning in Kubernetes for Fog Computing Applications', 2019 IEEE Conference on Network Softwarization (NetSoft), pp. 351-9.

[22] Concepts, Kubernetes viewed 25/05/2020, https://kubernetes.io/docs/concepts

[23] Horizontal Pod Autoscaler, Kubernetes viewed 25/05/2020, https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/

[24] Winarno, E., Hadikurniawati, W. & Rosso, R.N. 2017, 'Location based service for presence system using haversine method', 2017 International Conference on Innovative and Creative Information Technology (ICITech), pp. 1-4.

[25] Monawar, T., Mahmud, S.B. & Hira, A. 2017, 'Anti-theft vehicle tracking and regaining system with automatic police notifying using Haversine formula', 2017 4th International Conference on Advances in Electrical Engineering (ICAEE), pp. 775-9.

[26] User Guide 2019, vyos.io, viewed 26/05/2020, https://wiki.vyos.net/wiki/User_Guide