Middleware for resource sharing in fog computing with IoT applications

Hezekiah Samwini
Department of Computer Science
Edge Hill University
Ormskirk, United Kingdom
Samwinih@edgehill.ac.uk

Muhammad Awais
Department of Computer Science
Edge Hill University
Ormskirk, United Kingdom
Awaism@edgehill.ac.uk

Ella Pereira
Department of Computer Science
Edge Hill University
Ormskirk, United Kingdom
Pereirae@edgehill.ac.uk

Abstract—Fog computing is a new computing paradigm that extends cloud computing to make it more compatible with new Internet of Things applications. The aim of fog computing is to make the cloud capable of handling delay-sensitive applications, improve location-awareness, and reduce the volume of data sent to data centers for processing. In this paper, we propose a middleware for resource and load sharing for fog applications. Initial simulation results for some features of the middleware, conducted in iFogSim with a medical monitoring application for electroencephalogram (EEG) signals, are also presented.

Index Terms—fog computing, middleware, clustering, resource management, load balancing

I. Introduction

The increasing number of "smart" devices at the network edge presents a challenge for the current cloud computing architecture. In the current cloud architecture, end users make use of services by sending data from the edge through the network core to distant data centres owned and managed by cloud service providers (Amazon, Google, Microsoft, IBM etc.). While this model has worked well so far, the increasing number of smart or IoT devices at the edge poses challenges to this arrangement. First, smart devices produce high volumes of data (or Big Data) [1], that require processing. The processing of data and acting on the outcome is what makes smart devices smart [2]. This notwithstanding, the characteristics of smart devices (small form factors, little or no processing or storage resources and low power) mean that they cannot process the data they produce [3]. This leaves the task of processing Big Data to the cloud.

Moreover, for certain IoT services, processing in the cloud may not be useful due to performance, legal or security concerns. Applications in healthcare are a good example. For emergency applications, the delay in processing data in the cloud may be the difference between a patient receiving the right care or not. Additionally, in certain jurisdictions, processing personal data in a cloud data centre in another country or region is prohibited.

To address these challenges, Cisco proposed Fog Computing, as an extension of cloud computing, to complement the cloud by processing data closer to its source (at the network edge or core) to improve the user experience for delay-sensitive applications [4]. Fog Computing makes use of network devices, servers and other devices close to the user for the processing of user requests.

Abstraction is one of the key features contributing to cloud computing's success [5]. The ability to pool resources together within and across data centres makes it possible for the user to experience their expected quality of experience without any knowledge of disruptions or failures to nodes in the system. This kind of abstraction is possible in the cloud because data centres are often owned and managed by the same cloud service provider. However, in a fog computing setup, individual fog nodes may belong to a third party, such as a local ISP, or mobile carrier. They could be leased out to run a service on behalf of the cloud to improve the Quality of Service for users. Two challenges could arise from this. Firstly, the limited availability of resources on fog nodes means there must be alternative arrangements in the event of failure. The alternative cannot be at the cloud layer since this would defeat the purpose of processing close to the user. Secondly, since the location of services may change at any time, users must be able to locate the service they require when they need it without contacting the cloud – especially for emergency or delay-sensitive applications. To address these challenges, a middleware for the fog layer is proposed. Although middleware has been used in distributed systems to address the above challenges, its use in fog computing has not been fully investigated. This paper presents the design for a middleware for service discovery and scheduling at the fog layer. Results of simulations to compare the performance of the middleware's resource-sharing feature(clustering) with direct fog/edge processing and cloud processing are also presented.

II. Related Work

Middleware have been proposed to provide an abstraction layer to solve architecture mismatch problems associated with connecting IoT Systems to the cloud [6]. Middleware hides the complexity and heterogeneity of a system and makes it easier to develop applications for it [7]. The papers in [8]–[10] have introduced provisioning middleware for IoT clouds at the edge similar to fog computing. Their middleware has a distributed architecture

with components in the cloud as well as on gateway devices close to IoT devices.

The need for some form of middleware/abstraction layer in fog architectures has been discussed since the early days of fog computing. In [3], Bonomi proposed a fog abstraction layer to hide heterogeneity and manage resources at the fog layer. Nath et al. [11] also identifies the need for middleware at the fog layer to control network and other resources on fog nodes. Moreover, Aazam et al. [12] view the entire fog layer as a middleware for Cloud Computing. A few proposals for fog middleware have been made.

The works in [13], [14] present fog middleware architectures. Paper [13] presents Distributed IoT-Fog Architecture for Application Management (DIFAAM) to manage the life cycle of applications as they are run within the fog or cloud. Its goal is to ensure that application requirements would be met by fog devices before assigning tasks to them. Pore et al. [14] also propose an architecture for fog and edge middleware. Theirs focuses on task scheduling and data collection in mobile fog environments.

Works by Nader et al. [15]–[17] present a Service-Oriented Middleware for fog computing. In [16], a Service-Oriented Middleware approach is adopted for a smart city. The middleware abstracts system resources as services which are made accessible to devices across all layers of the system. The authors implement their middleware for a cyber-physical system in [17]. The Service-Oriented approach makes it possible to add new services after deployment. Also, a Service Oriented Middleware provides flexibility for large-scale IoT applications [15].

Paper [18] presents a Fog-based middleware for distributed cooperative data processing at the fog layer. In their proposed middleware fog nodes have two modes; they either work together on a task or work independently. Their system is implemented in a specific application - subsurface imaging and monitoring.

Shekhar et al. [19] use middleware for task offloading in a mobile IoT environment. Their middleware manages resources across all layers of the fog architecture with the goal of ensuring that service-level objectives are met even when edge devices are mobile. Their proposed solution, however, requires prior knowledge of the user's movement which is not practical in real-life scenarios. Other middleware have been designed for privacy and security in fog systems [20], [21]

Middleware proposed for fog computing have focused on specific application scenarios and are mostly unsuitable for other fog applications. In this paper, a middleware to provide abstraction, and enable resource sharing and load balancing at the fog layer is presented.

III. Architecture and Problem Description

The middleware uses the three-layer fog architecture shown in Fig. 1. The bottom layer or End Devices Layer has IoT, smart and user devices which produce data and

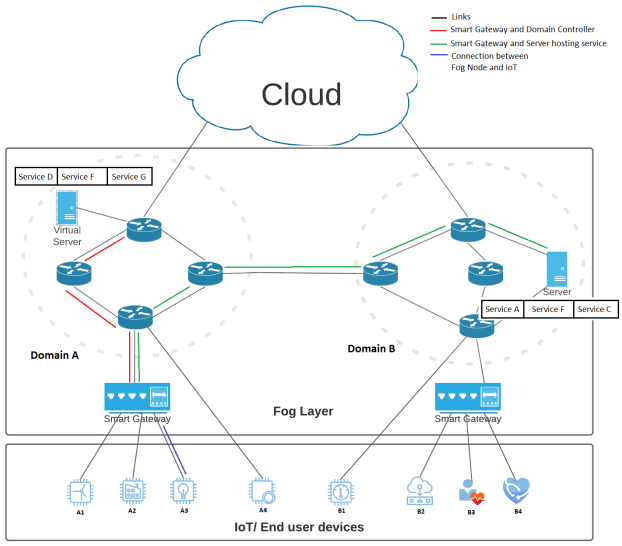

Fig. 1. Problem Scenario

execute instructions from applications. Devices at this layer require access to services from a higher layer (fog or cloud). Also, data produced at this layer vary in volume, speed, and longevity. These variations have implications for the nature of the processing required. Additionally, applications have different Quality of Service requirements that must be met.

In the scenario presented in Fig. 1, device A3 wants to access service C which is not available on any device within the domain it is connected to. The service is however on a virtual machine on a fog node in a nearby domain (domain B). How would device A3 access the service from the fog node without connecting to the cloud? The proposed approach, using middleware is described in the following:

- Device A3 connects to the fog node closest to it (Smart Gateway) with a message containing the service requested (Service C and the parameters or inputs).
- The Smart Gateway looks up the lists of services available within its domain and does not find Service C.
- The Smart Gateway sends a request to the domain controller for Service C.
- The domain controller looks up services available on neighbouring domains and finds service C in domain B. The controller sends a response back to the Smart Gateway with the address to service C and how to reach it.
- The Smart Gateway sends a request to the server requesting access to the server. The server responds with a service description.
- Smart Gateway changes device C's message to a service request to the server and sends outputs back to device C.

Depending on the performance of application C, IoT devices may request a better performance (lower latency). In this case, Smart Gateway may send a request to the domain controller requesting that Service C be hosted within the domain. If there are resources available within the domain to host service C, the domain controller will send a request to the fog orchestrator (in the cloud) to send the service to the appropriate device. Alternatively, if the connection between the Smart Gateway and the hosting service is still active, the virtual machine can be migrated from the hosting server to another server in the Smart Gateway's domain.

In the above description, the role of the middleware spreads across the devices at the fog layer. Middleware is responsible for the interactions between devices in the same domain. The middleware also converts IoT request messages (Message-Oriented Communication) into Service requests (Service-Oriented Communication) between the Smart Gateway and the Server (fog-fog interaction).

Also, the middleware interacts with the domain controller and keeps a record of services available within its domain. The record is updated frequently as new services are introduced within the domain. Within the domain fog nodes (network devices, servers etc.) host and give up services based on the availability of resources, demand etc.

IV. Proposed Middleware

Fog nodes may be any device between the cloud and the end user. Computational resources at different levels of the network (gateway, access, core) are made available for pre-processing or semi-processing of IoT data. Different entities own and manage fog nodes. Also, fog nodes may be organised into domains or run as stand-alone systems. Additionally, the computational resources made available by nodes vary and may be increased or reduced depending on availability and workload. Consequently, a fog node may not always have the resources needed to process IoT data sent to it.

The proposed middleware is hosted on fog nodes and interacts with Iot devices, the cloud and other fog devices. Fog nodes may be any device with computational resources at any level of the network – from consumer devices to dedicated servers. Consequently, their resources are not comparable to the large computational resources that are available in the cloud.

Since fog nodes are managed and owned by different entities, they may run different operating systems or platforms and may not be interoperable. The role of the middleware is to make it possible for heterogeneous fog nodes to interact with each other for resource sharing to improve the reliability of the system.

Services are applications or parts of applications which provide a service to IoT devices. Services perform single functions and may be chained to form a complete application.

Users/IoT devices request services from the cloud or fog devices. They access services by sending their user ID, service ID and required data to the nearest fog node. IoT devices are resource constrained.

The cloud is resource-rich and provides services for end users and IoT devices. All services are available in the cloud; however, some services are fully or partially shared with fog devices to improve their performance.

The proposed middleware is designed for communication among fog devices at the same level, lower or upper levels. The interaction between fog nodes is mainly for information exchange, request forwarding and load balancing. Fog nodes share data on the services they are currently hosting and their resource availability. They also forward IoT requests to other fog nodes for processing and offload tasks to other fog nodes when overloaded. Moreover, the middleware is designed to also communicate with the cloud. Communication with the cloud is for orchestration, exchanging end-user information, sending data for further processing, etc. Fog nodes perform various tasks for the cloud. This is an important distinction between fog computing and other edge-based paradigms such as edge computing. In fog computing, fog nodes do not work independently of the cloud. The role of the middleware is to interact with the cloud to define and set up the expected role of the node. The middleware features a request handler, task scheduler, service registry, service discovery and communication modules.

Algorithm 1 shows the algorithm for request handling. Requests from IoT devices are first received at the request handler. The request handler looks up the service requested by the IoT device in the service registry. If the service is hosted on the fog node and the node has the resources to run it, the data will be processed on the fog node.

Algorithm 1 Request Handling Algorithm

```
Require: IoT Request
Ensure: Match Request to Service
Retrieve requested service
if service is hosted on the fog node then
    if node can meet the request then
        Assign task to this node
    end if
else if Service NOT hosted on fog node then
    Lookup Service in Service Registry
    if service is in Service Registry AND Host Node is free then
        Send request to Host Node
    else
        Send Service Lookup Request to Domain Controller
    end if
end if
```

If the request cannot be processed on the fog node, it

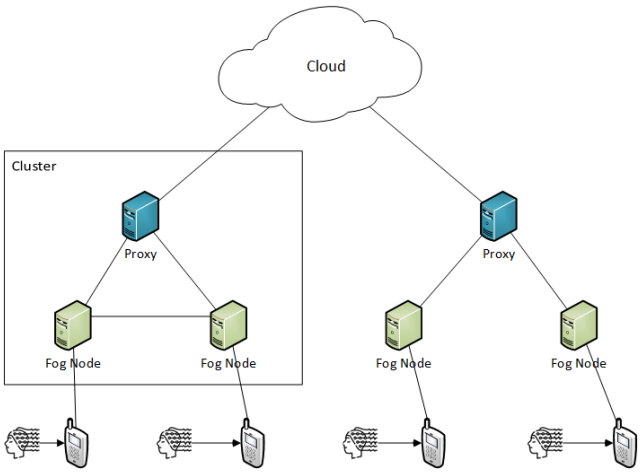

Fig. 2. Architecture used in the simulation

searches for another fog node to send the request to. First, the handler will look up the service in the Service Registry, if the service is hosted on a known node (a node listed on the registry), the request is forwarded to that node.

The middleware maintains a list of services and fog nodes which are hosting them. The list is updated regularly as services are removed and added regularly by fog nodes based on demand and availability of resources.

A Service Discovery module searches for new services when they are requested by IoT devices. New services are discovered through other fog nodes, a fog controller, or the cloud. When a fog node receives high volumes of requests for a service it does not host, it may request to host that service.

The task scheduler prioritises tasks that are processed on the fog node. Tasks are prioritised based on QoS requirements and/or Service Level Agreements. The scheduler keeps track of resources available on the node and allocates tasks to them.

V. Simulation

To test some of the functionalities of the proposed middleware, we run simulations using iFogSim [22], a popular java-based simulation tool for fog computing. For these simulations, we modelled an Electroencephalography (EEG) application which receives EEG signals from a user, processes the signals (on fog nodes or in the cloud) and sends feedback to the user. Fig. 2 shows the setup for the simulation.

Wireless Body Area networks make long-term physiological monitoring possible outside of the hospital [23]. Wireless Electroencephalography(WESN) provides the possibility for early detection, monitoring and treatment of diseases such as epilepsy, Parkinson's and Alzheimer's. The WESN is made of an array of EEG sensing nodes each of which consists of an electrode array, signal processing unit and a transceiver for communication. Amplitude integrated EEG may be useful in monitoring infants

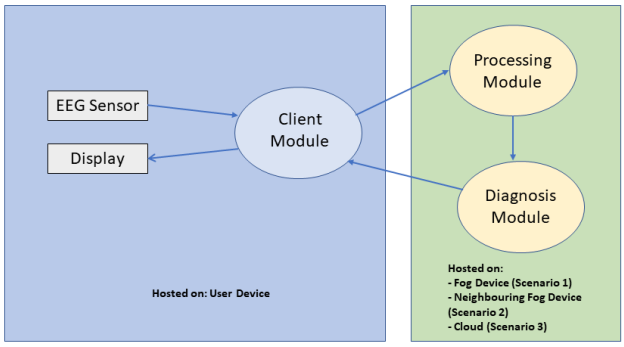

Fig. 3. Application Model for the simulation

to predict the possibility of future or subsequent brain damage [24]. We modelled an EEG application with three modules (a client module, pre-processing and diagnosis). The EEG sensor generates signals up to 200Hz which is within the range for EEG signals monitoring. As EEG data is often noisy and often affected by artefacts on the patient or the environment, pre-processing is necessary to remove the noise from the signal [25]. Signals are cleaned and filtered by the client and pre-processing module before being analysed by the diagnosis module.

To evaluate the performance of the application with the introduction of clustering, simulations were run for three scenarios. In the first scenario, modules are hosted on a single fog node (with no collaboration with other fog nodes). In the second scenario, the user request is forwarded to another fog device with more resources. The third scenario runs the application completely in the cloud. Data is sent from one node to another after processing and the feedback or actuation signal is sent back to the user. The Sensor and Display may be an EEG device and mobile phone attached to the patient or user. Fig. 3 shows the application design.

VI. Results and Discussion

Delay: Fig. 4 illustrates the delay for the EEG application as the number of users per fog node increases for the three scenarios. As data must travel across multiple nodes, and with increasing network latency towards the cloud, the cloud-based scenario has the highest latency. Also, processing on the fog node closest to the user has a lower latency compared to processing on another fog in the same cluster. However, when the number of users increased to 5, processing on another fog node in the same cluster produced a higher latency since that device had more processing resources compared to the fog device that first received the request. With higher computational and storage resources, the third-party fog node provides better latency because the delay from scheduling the request is higher than the propagation delay. The delay involved in processing for multiple users on the closest edge device

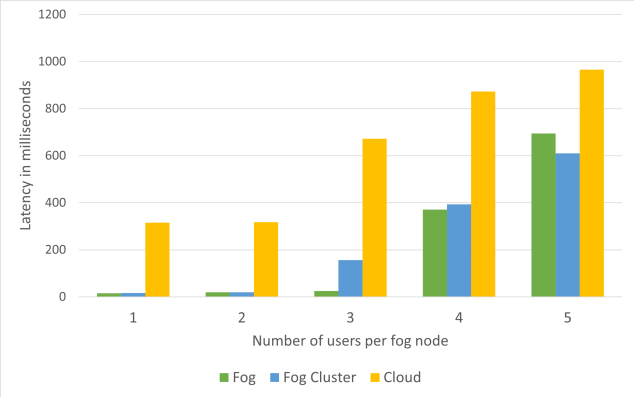

Fig. 4. Comparison of application loop latency for the three scenarios

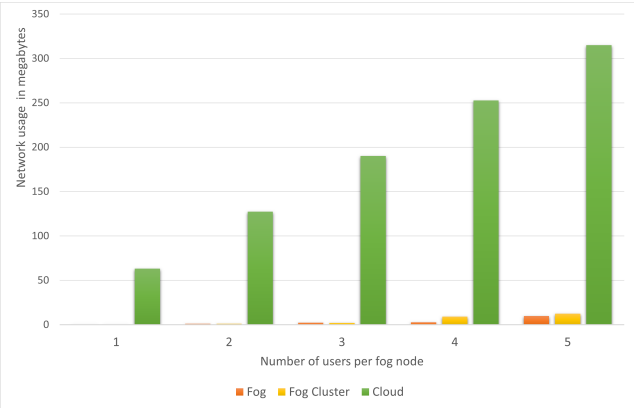

Fig. 5. Comparison of network usage for the three scenarios

grew higher than the network delay from forwarding to another device in the same cluster.

Network usage: The network usage for the application in the three scenarios is shown in fig. 5. Network usage increases significantly when processing is done in the cloud. As a result of traversing multiple nodes and links and the increased latency link to the cloud, network usage in the cloud scenario is much greater compared to processing on fog nodes. This also shows the scalability of processing at the fog layer [22]. Moreover, the usage for processing on a fog node within the same cluster is processing on the node closest to the user. Consequently, as IoT requests from users increase, increasing the number of fog nodes available to process the requests, and load sharing among fog devices, would ensure applications remain reliable without resorting to forwarding to the cloud.

Energy Consumed: Fig. 6 shows the energy consumption of the different devices in the simulation and the data centre for five users. The energy usage for fog devices is the same when requests are run at the fog layer, but reduces to the same level as the proxy when requests are run in the cloud. When processing is done in the cloud, the energy usage of fog nodes is the same as that of the

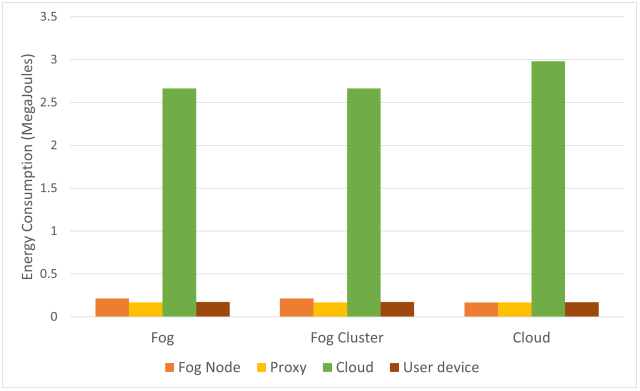

Fig. 6. Comparison of energy consumption for devices in each scenario

proxy since they all act as network devices to only forward data to nodes above or below them. Energy consumed by user devices remains the same for all scenarios because they process the client module of the application in all scenarios. The cloud data centre's consumption remains significantly higher because of the number of resources available and the high energy use even when running on idle power. It increases further when the processing is done in the cloud.

VII. Conclusion and Future Work

Fog Computing makes resources available for processing user requests close to the network edge where the data is produced. Fog reduces the latency of IoT applications and reduces the load on the cloud by processing data closer to the user instead of in the cloud. We have presented a middleware to enable the abstraction of resources at the fog layer and make applications more reliable. Results from initial simulations show that the pooling of resources at the fog layer can improve the reliability of applications as user requests increase. In the future, we will be investigating other aspects of the middleware related to the communication of nodes and the scheduling of multiple tasks with different requirements.

References

[1] C. Doctorow, "Big data: Welcome to the petacentre," Nature, vol. 455, no. 7209, pp. 16–21, sep 2008. [Online]. Available: http://www.nature.com/doifinder/10.1038/455016a

[2] A. Yassine, S. Singh, M. S. Hossain, and G. Muhammad, "IoT big data analytics for smart homes with fog and cloud computing," Future Generation Computer Systems, vol. 91, pp. 563–573, feb 2019. [Online]. Available: https://doi.org/10.1016/j.future.2018.08.040https://linkinghub.elsevier.com/retrieve/pii/S0167739X18311099

[3] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, "Fog Computing: A Platform for Internet of Things and Analytics," in Big Data and Internet of Things: A Roadmap for Smart Environments, N. Bessis and C. Dobre, Eds. Springer, Cham, 2014, pp. 169–186. [Online]. Available: http://link.springer.com/10.1007/978-3-319-05029-4_7

[4] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in Proceedings of the first edition of the MCC workshop on Mobile cloud computing - MCC '12. New York, New York, USA: ACM Press, 2012, p. 13. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2342509.2342513

[5] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud Computing and Grid Computing 360-degree compared," Grid Computing Environments Workshop, GCE 2008, 2008.

[6] A. Farahzadi, P. Shams, J. Rezazadeh, and R. Farahbakhsh, "Middleware technologies for cloud of things: a survey," Digital Communications and Networks, vol. 4, no. 3, pp. 176–188, 2018.

[7] A. Carrega, M. Repetto, P. Gouvas, and A. Zafeiropoulos, "A Middleware for Mobile Edge Computing," IEEE Cloud Computing, vol. 4, no. 4, pp. 26–37, jul 2017. [Online]. Available: http://ieeexplore.ieee.org/document/8066002/

[8] S. Nastic, S. Sehic, D.-H. Le, H.-L. Truong, and S. Dustdar, "Provisioning Software-Defined IoT Cloud Systems," in 2014 International Conference on Future Internet of Things and Cloud. IEEE, aug 2014, pp. 288–295. [Online]. Available: http://ieeexplore.ieee.org/document/6984208/

[9] M. Vogler, J. Schleicher, C. Inzinger, S. Nastic, S. Sehic, and S. Dustdar, "LEONORE – Large-Scale Provisioning of Resource-Constrained IoT Deployments," in 2015 IEEE Symposium on Service-Oriented System Engineering, vol. 30. IEEE, mar 2015, pp. 78–87. [Online]. Available: http://ieeexplore.ieee.org/document/7133516/

[10] S. Nastic, H.-L. Truong, and S. Dustdar, "A Middleware Infrastructure for Utility-Based Provisioning of IoT Cloud Systems," in 2016 IEEE/ACM Symposium on Edge Computing (SEC). IEEE, oct 2016, pp. 28–40. [Online]. Available: http://ieeexplore.ieee.org/document/7774351/

[11] S. B. Nath, H. Gupta, S. Chakraborty, and S. K. Ghosh, "A Survey of Fog Computing and Communication: Current Researches and Future Directions," arXiv, no. April, apr 2018. [Online]. Available: http://arxiv.org/abs/1804.04365

[12] M. Aazam and E.-N. Huh, "Fog Computing: The Cloud-IoTIoE Middleware Paradigm," IEEE Potentials, vol. 35, no. 3, pp. 40–44, may 2016. [Online]. Available: http://ieeexplore.ieee.org/document/7466912/

[13] P. Bellavista, L. Foschini, N. Ghiselli, and A. Reale, "MQTT-based Middleware for Container Support in Fog Computing Environments," in 2019 IEEE Symposium on Computers and Communications (ISCC). IEEE, jun 2019, pp. 1–7. [Online]. Available: https://ieeexplore.ieee.org/document/8969615/

[14] M. Pore, V. Chakati, A. Banerjee, and S. K. S. Gupta, "Middleware for Fog and Edge Computing: Design Issues," in Fog and Edge Computing: Principles and Paradigms, R. Buyya and N. S. Srirama, Eds. Hoboken, NJ, USA: John Wiley Sons, Inc., jan 2019, pp. 123–144. [Online]. Available: http://doi.wiley.com/10.1002/9781119525080.ch6

[15] J. Al-Jaroodi, N. Mohamed, I. Jawhar, and S. Mahmoud, "CoTWare: A Cloud of Things Middleware," in 2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW). IEEE, jun 2017, pp. 214–219. [Online]. Available: http://ieeexplore.ieee.org/document/7979819/

[16] N. Mohamed, J. Al-Jaroodi, I. Jawhar, S. Lazarova-Molnar, and S. Mahmoud, "SmartCityWare: A Service-Oriented Middleware for Cloud and Fog Enabled Smart City Services," IEEE Access, vol. 5, pp. 17 576–17 588, 2017.

[17] N. Mohamed, S. Lazarova-Molnar, I. Jawhar, and J. Al-Jaroodi, "Towards Service-Oriented Middleware for Fog and Cloud Integrated Cyber Physical Systems," in 2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW). IEEE, jun 2017, pp. 67–74. [Online]. Available: http://ieeexplore.ieee.org/document/7979797/

[18] J. Clemente, M. Valero, J. Mohammadpour, X. Li, and W. Song, "Fog computing middleware for distributed cooperative data analytics," in 2017 IEEE Fog World Congress (FWC). IEEE, oct 2017, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/8368520/

[19] S. Shekhar, A. Chhokra, H. Sun, A. Gokhale, A. Dubey, and X. Koutsoukos, "URMILA: A Performance and Mobility-Aware Fog/Edge Resource Management Middleware," in 2019 IEEE 22nd International Symposium on Real-Time Distributed Computing (ISORC). Valencia, Spain: IEEE, may 2019, pp. 118–125. [Online]. Available: https://ieeexplore.ieee.org/document/8759356/

[20] A. M. Elmisery, S. Rho, and D. Botvich, "A Fog Based Middleware for Automated Compliance With OECD Privacy Principles in Internet of Healthcare Things," IEEE Access, vol. 4, pp. 8418–8441, 2016. [Online]. Available: http://ieeexplore.ieee.org/document/7805329/

[21] B. Mukherjee, S. Wang, W. Lu, R. L. Neupane, D. Dunn, Y. Ren, Q. Su, and P. Calyam, "Flexible IoT security middleware for end-to-end cloud–fog communication," Future Generation Computer Systems, vol. 87, pp. 688–703, oct 2018. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0167739X17311470

[22] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, "iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments," Software: Practice and Experience, vol. 47, no. 9, pp. 1275–1296, sep 2017. [Online]. Available: http://doi.wiley.com/10.1002/spe.2509

[23] A. Bertrand, "Distributed Signal Processing for Wireless EEG Sensor Networks," IEEE Transactions on Neural Systems and Rehabilitation Engineering, vol. 23, no. 6, pp. 923–935, 2015.

[24] L. Hellstrom-Westas, I. Rosen, and N. W. Svenningsen, "Predictive value of early continuous amplitude integrated EEG recordings on outcome after severe birth asphyxia in full term infants." Archives of Disease in Childhood - Fetal and Neonatal Edition, vol. 72, no. 1, pp. F34–F38, jan 1995. [Online]. Available: https://fn.bmj.com/lookup/doi/10.1136/fn.72.1.F34

[25] A. Delorme, "EEG is better left alone," Scientific Reports, vol. 13, no. 1, p. 2372, feb 2023. [Online]. Available: https://www.nature.com/articles/s41598-023-27528-0