Mustansiriyah Journal of Pure and Applied Sciences

www.MJPAS.com

iFogger: A New Framework to Simulate Fog Computing Resource Placement inside OMNeT++ Environment

B.A. Hussein [1,*] and S.H. hashem [2]

[1] Computer Center, University of Babylon, Iraq (balasem@uobabylon.edu.iq)
[2] Computer Sciences Department, University of Technology, Iraq.
(110015@uotechnology.edu.iq)

A B S T R A C T

The capabilities of the cloud are extended to intermediate network devices by fog computing, providing computational and storage resources. This extension enables the execution of applications closer to edge devices and end-users by deploying services on these intermediate devices. The performance of the fog architecture is significantly impacted by the placement of these services. In this study, iFogger, a fog computing simulator, is proposed to analyze the design and deployment of applications using customized and dynamic strategies. To achieve this, the relationships among deployed applications, network connections, and infrastructure characteristics are modeled using complex network theory. This allows the integration of topological measures in dynamic and customizable strategies, including the placement of application modules, workload location, and path routing and scheduling of services. The iFogger simulator is built on top of the OMNeT++ network simulator. A comparative analysis is performed to assess the efficiency and convergence of results obtained with our simulator compared to the widely referenced iFogSim. The results show that iFogger exhibits better performance in terms of simulation time convergence.

Keywords: *Fog Computing; IoT; Resource placement; QoS.*

1. Introduction

The increased number of IoT devices lead to numerous amounts of data transferring between different applications and devices [1], [2]. This data is processed by their consumers; industries or individuals to derive different types intellectual knowledge or decisions. Alongside, a rapid increase in the number of establishments have already lift their critical data to the cloud which provide pay-per-use, scalability, and availability [3]. Cloud computing deliver several powerful services such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS), that goes along with current trend for Everything as a Service (XaaS) [4]. The huge pulses of data generated from millions of harmonic and diverse IoT entities are far from being smoothly consumed by the current convenient cloud, in consequence a lot of latencies will be generated. Dissolving this problem was done by the means of Fog Computing, which provide enough computing, storage and networking to extend the range of the cloud and act as a mediator to bring the cloud

nearer to the IoT entities [5]. Table 1 illustrates the main features of Cloud, Fog and IoT environments.

Table 1. Main features of Clou/Fog/IoT environments

Aspect	Cloud	Fog	IoT
Location of resources	Centralized datacenters	Distributed edge devices	Distributed end devices
Proximity to users	Distant	Close to edge	Closest to end users
Latency	Higher	Lower	Ultra-low
Network usage	High	Reduced	Efficient
Data processing	Centralized	Distributed	Local
Scalability	High	Medium	High
Reliability	Dependent on datacenters	Distributed and redundant	Highly variable
Connectivity	Broadband networks	Multi-hop wireless	Wireless and limited
Workload types	Data-intensive	Latency-sensitive	Real-time and low-power
Applications	Generalized	Industry-specific	Diverse
Resource flexibility	Limited	Dynamic	Dynamic and adaptive

The functionality of the Fog will increase the efficiency and performance by rerouting the information gathered by the IoT devices to a compute structure like the edge for processing and storage, instead of passing them over directly to the cloud. Consequently, the latency and bandwidth requirements will be greatly reduced [6]. As a result, this integrations between IoT and Fog produce what is called Fog as a Service (FaaS) [7], Figure 1. This service enables deployment and manage scalable resources to Fog service providers in any volume and capacity [8]. The distributed nature of Fog environment and the diverse types of devices contributing in generating network flow in addition to, generally, wide geo-coverage of these devices makes resource placement a difficult and challenging task [9]. The study of the feasibility of a placement algorithm needs to be investigated in virtually two approaches; a physical arrangement, alternatively researchers use network simulation tools. In addition, a hybrid method can be used to augment logical nodes behaviors with physical implementation [10].

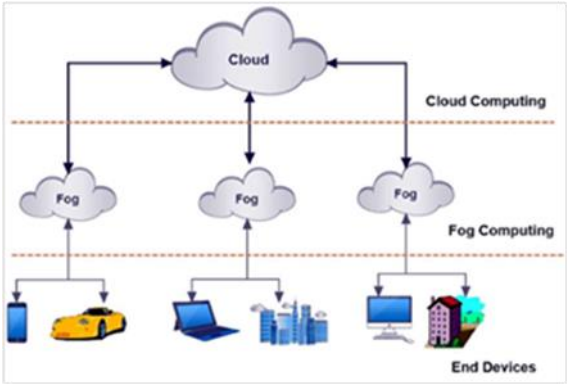

Figure 1. Fog computing environment [31]

2.Fog/Edge Computing Simulators

The Fog simulators taken into consideration as related frameworks to this work are iFogSim [11], [12], FogNetSim++ [13], EdgeCloudSim [14], YAFS [15], MyiFogSim [16], MobFogSim [17], ECSNeT++ [18], and DISSECT-CF-Fog [19], [20]. The common aspect between those simulators is they generally present the way Fog environment works to be further investigated by researchers. Table 2 shows comparison of architectural features. The table shows the environment simulated which include IoT, Edge, Fog, and Cloud to emphasize the scope of the simulator implementation. The programing language of the simulator shows a dominant of Java language over most of the Fog simulators with exception of FogNet++ and ECSNet++ which written in C++ programing language, and YAFS written in Python programing language. The core simulators that were written in Java generally use the commonly used cloud simulator CloudSim and implemented as extensions. In addition, iFogSim was used as core simulator for some Fog simulators which itself is an extension of CloudSim. As for YAFS, it uses the PySim library in Python as the core simulator.

Table 2. Comparison between selected simulators in terms of environment simulated, programing language and core simulator.

Simulator	Environment	Language	Core simulator
AFS	Fog	Python	PySim
EdgeCloudSim	Edge	Java	CloudSim
iFogSim	Fog	Java	CloudSim
MyiFogSim	Fog	Java	iFogSim
MobFogSim	Fog	Java	iFogSim
DISSECT-CF-Fog	Fog	Java	DISSECT-CF
FogNETSim++	Fog	C++	OMNET++
ECSNet++	Fog	C++	OMNET++
***iFogger* (*this work*)**	***Fog***	***C++***	***OMNET++***

Another dimension to characterize Fog simulators is the technical properties of the simulator concerning user and developers of Fog simulators. The technical aspects (as in Table 3, and Table 4) include features like validation of simulations (i.e., using real datasets and benchmarking) and the Fog environment specific scenarios (implementation of capabilities like customizable mobility, nodes clustering, and support of microservices), in addition to the deployment of cost models, energy consumption modelling, resources administration, results generation, and in simulation time visualization and animation. Generally, the adaptation of any of these features is task specific depending on the main goal of the simulator.

Table 3. Fog Simulators Main Specifications

Simulator	Using real data	Benchmarking	GUI	Mobility Modelling	Customizable Mobility	Nodes Clustering	Micro-services	Topology
YAFS	Y	Y	Y	Y	N	Y	Y	Fog/IoT
EdgeCloudSim	N	Y	N	Y	N	N	N	Edge
iFogSim	Y	Y	Y	Y	Y	Y	Y	Cloud/Fog
MyiFogSim	Y	Y	Y	Y	N	N	N	Fog/Edge
MobFogSim	Y	Y	Y	Y	N	N	Y	Fog/Edge
DISSECT-CF-Fog	N	N	N	Y	N	N	N	Cloud/Fog
FogNETSim++	N	N	Y	Y	Y	N	N	Cloud/Fog
ECSNet++	Y	Y	Y	Y	Y	N	Y	Cloud/Edge
***iFogger* (*this work*)**	***Y***	***Y***	***Y***	***Y***	***Y***	***Y***	***Y***	***Cloud/Fog/IoT***

Table 4. Technologies Implemented in Fog Simulators

Simulator	Cost Model	Energy Model	Resource Management	Simulation Visualizations	Simulation Animation	Results Generations	Results Visualizations
YAFS	N	Y	N	N	N	Y	N
EdgeCloudSim	N	Y	Y	N	N	Y	N
iFogSim	Y	Y	Y	N	N	Y	N
MyiFogSim	Y	Y	Y	N	N	Y	N
MobFogSim	Y	Y	Y	N	N	Y	N
DISSECT-CF-Fog	N	Y	Y	N	N	Y	N
FogNETSim++	Y	Y	N	Y	Y	Y	Y
ECSNet++	Y	Y	N	Y	Y	Y	Y
***iFogger* (*this work*)**	***Y***	***Y***	***Y***	***Y***	***Y***	***Y***	***Y***

3.iFogger Implementation

Fog computing environment simulators differ in implementation and purpose [21], as described in section 2. This section will describe the implementation details of the proposed simulator (iFogger). The iFogger simulator is conceptually built from the widely used Fog simulator, iFogSim and implemented on top om a commonly used network simulator OMNeT++. This approach enabled us to benefit from the solid Fog concepts adopted by iFogSim and transfer them to pure C++ code inside OMNeT++ which offer important facilities like a sophisticated graphical user interface (GUI) to build networks and built-in reports generator.

The rest of this section will compare iFogger with iFogSim as the main source of our simulator. Then, a comparision with Fog simulators already built using OMNeT++, namely FogNetSim++ and ECSNeT ++.

3.1 iFogger and iFogSim

The main motivation for us to implement iFogSim on top of OMNeT++ is the richness of features embedded in iFogSim and additionally the wide use of this simulator in academic papers concerning different disciplines of Fog computing implementations (at the time of writing the main transcript of iFogSim [12] was cited by more than 1350 academic papers and publications). Another reason is its credibility in simulations validation against real network traffic in addition to the wide variety of resource management algorithms[22]. iFogSim is very scalable when comes to network design in both vertical (number of computing layers) and horizontally (number of entities per computing layer. The realization of a Fog application in iFogSim is to define the flow of entities through the fog distributed environment, starting from sensors sending tuples to the fog devices and then processed them to get results to be sent to actuators[23]. The task of orchestration the processing flow is dominated by a Fog controller (i.e., Fog broker) to handle resources management[24], [25]. Generally, a Fog controller audit the placement of VMs according to the network's Fog devices' computing constraints and application(s) requirements[26]. Accordingly, Fog device allocate the required resources (e.g., CPU, ram, and bandwidth) for these VMs and control the scheduling of requests[27].

Basically, preparing an application to be executed on iFogSim requires manual setting for devices or using a GUI tool to visualize the network devices, which are limited to Sensor, Device, Actuator, and the links between them, Figure 2. While in iFogger, as it's an OMNeT++ based simulator, visualizing the network settings will be as simple as drag-and-drop of iFogger entities and any applicable OMNeT++ entity, in form of NED files (network definition files), Figure 3.

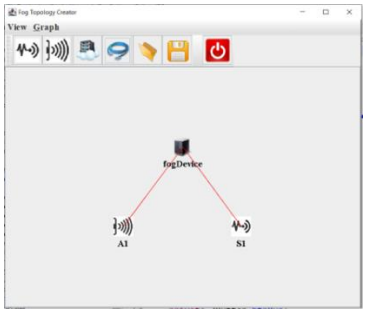

Figure 2. iFogSim GUI window

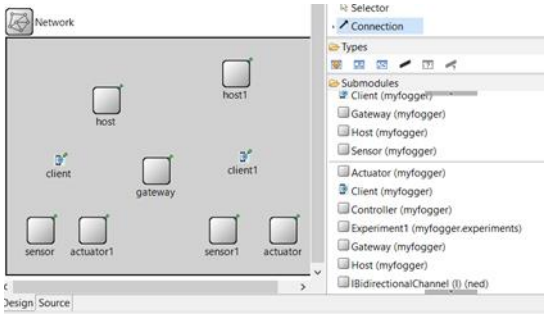

Figure 3. OMNeT++ Network design window

Communication channel delays in iFogSim is simulated using the edge latency property or events scheduling delays (for propagation delays). While in iFogger (baring OMNeT++) communication channels are modules can be used to simulate different types of communication delays in addition their classes can be derived to create any unimplemented channel behavior.

Typically, when simulating an environment like Fog computing the main purpose is to get results as basis to simulating insights or comparisons. iFogSim provide textual results for tuple flow and computing resources constraints and a final summary of performance, these results can be manually converted to useful outcomes. The results in iFogger are represented via the results engine implemented in OMNeT++ to be automatically represented graphically (such as line graphs and histograms) and statistical summaries to be exported in various common file formats (like CSV files and Excel files). In addition, OMNeT++ gives the ability to process simulations results using Python for further data analysis.

3.2 iFogger and OMNeT++ Fog Simulators

iFogger was designed as a framework to simulate Fog environment on top of OMNeT++ to simulate the performance of different algorithms for resource placement algorithms. Hence, to compare it with other OMNeT++ based simulators from the same scope two simulators are selected (i.e., FogNetSim++ [13] and ECSNet++ [18]).

FogNetSim++ is an open-source Fog environment simulator built on top of OMNet++ to simulate Fog networks and nodes [28]. Static along with dynamic devices can be used to simulate mobility and Fog network communication protocols, such as Message Queue Telemetry Transport (MQTT) [29] and Constrained Application Protocol (CoAP) [30]. Despite the limited support [31], device handover is a major contribution of the FogNetSim++ simulator. This simulator has two dependencies; OMNeT++ v4.6 and the high-level library INET v3.3.0. The internal structure and communication logic for Fog nodes is taken from INET without any significant alteration. In addition, mobility modeling and energy consumption models are also taken from INET. FogNetSim++ implements MQTT [29] as the Fog network communication protocol to simulate IoT layer messages to higher layers (Edge, Fog and Cloud) via its one hop access points. A central controlling node (Broker) is responsible for managing compute requests to maintain the Service License Agreement (SLA) [32] between the user and the service provider. This simulator lacks virtual machine migration which is necessary to achieve better load balancing between over-utilized nodes and under-utilized nodes[33].

ECSNet++ [18] suggested as a framework to simulate distributed stream processing (DSP) applications which built on top of OMNeT++ network simulator. The developers claimed that when there are sufficient calibrations to the network placement topology, the simulator can model real-world scenarios. Another feature, its ability to foresee the behavior and measure the performance of large-scale distributed applications run on multi-core, multi-thread CPUs. ESCNet++ evaluate various metrics like network environment delays (end-to-end, processing, and network) in addition to energy consumption evaluations. Placement plan in this simulator requires a special XML template to specify the placement requirements of the network devices (the template is provided with the framework, and a simple example is also provided which span over 126 lines). The placement plan is a

requirement for the simulation to run. In contrast to iFogSim and FogNetSim++, this simulator does not support SLA satisfaction. This simulator requires three dependencies; OMNeT++ v5.1, INET v3.6.0 [34] and TinyXML2 library to parse the placement plan XML document.

To emphasize the points of difference between iFogger and those two Fog simulators we can mention that in terms of virtual machines both simulators lack the capability to perform VM migration. In addition, user license agreement satisfaction is not considered in ECSNet++. The heavy dependency on INET library in OMNeT++ for both FogNetSim++ and ECSNet++ couples these simulators to the version used by the user because they cannot run their simulations unless the versions requirements met. In addition to the tinyXML2 library required by ECSNet++ to load placement plans.

Virtual machines migration is fully implemented in iFogger and works according to load balancing requirements for the particular host. The dependency issue noticed on the mentioned fog simulators are not present in iFogger. The development environment for iFogger is only the core of OMNeT++ simulator without any INET dependency or any OMNeT++ version specific feature. That means, once the user has a version of OMNeT++ IDE they can start developing their Fog environment, that elevates the usability of iFogger simulations, see Table 5.

Table 5. Fog Simulators Capabilities and Features

Simulator	VM Migration	OMNeT++ Version Independent	INET Version Independent	External Library Independent	SLA Support
iFogSim	Y	-	-	N	Y
FogNetSim++	N	N	N	Y	Y
ECSNet++	N	N	N	N	N
iFogger	***Y***	***Y***	***Y***	***Y***	***Y***

4. Simulation Experiment

We conducted a comparison between two simulators using the first case study from the iFogSim paper, specifically the EGG Tractor Beam game application. This application comprises three modules: client, concentration, and coordinator. The experiment deploys these modules in a hierarchical three-based topology, where a cloud entity is connected to a gateway that links all fog devices, Figure 4. The network can be scaled by generating multiple subgroups from the gateway device.

Two placement strategies were analyzed: a cloud-only placement where all modules are deployed in the cloud entity, and an edge policy where the modules are deployed in fog devices. We focused on analyzing execution time while varying the number of fog nodes (4, 8, 12, and 16).

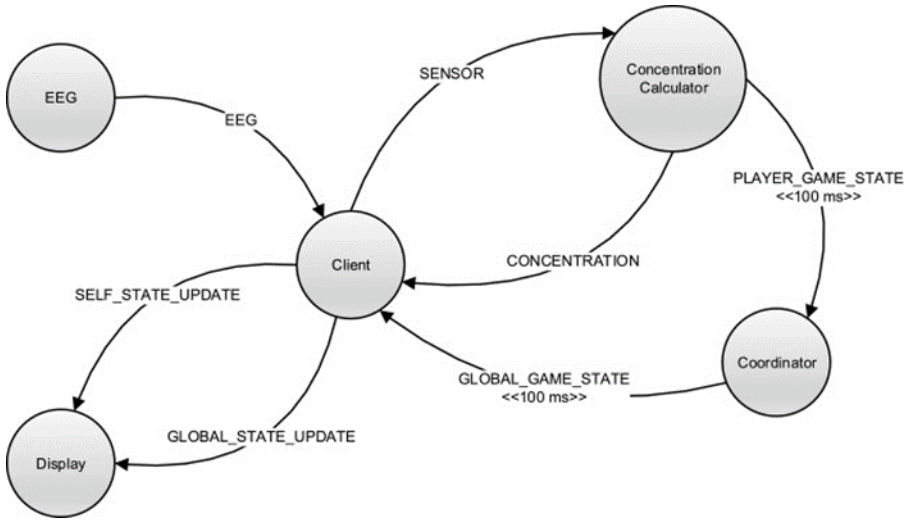

Figure 4. Logical flow of modules and messages for the EGG example application.

The simulation was executed on a machine with i7-core running at 3.745 GHz with 16 GB RAM. Figure 5 shows the execution time for both policies as the number of fog nodes increases. The blue lines represent the results of iFogSim, and the magenta lines represent iFogger. Sloid lines correspond to the cloud policy, and dashed lines represent the edge policy.

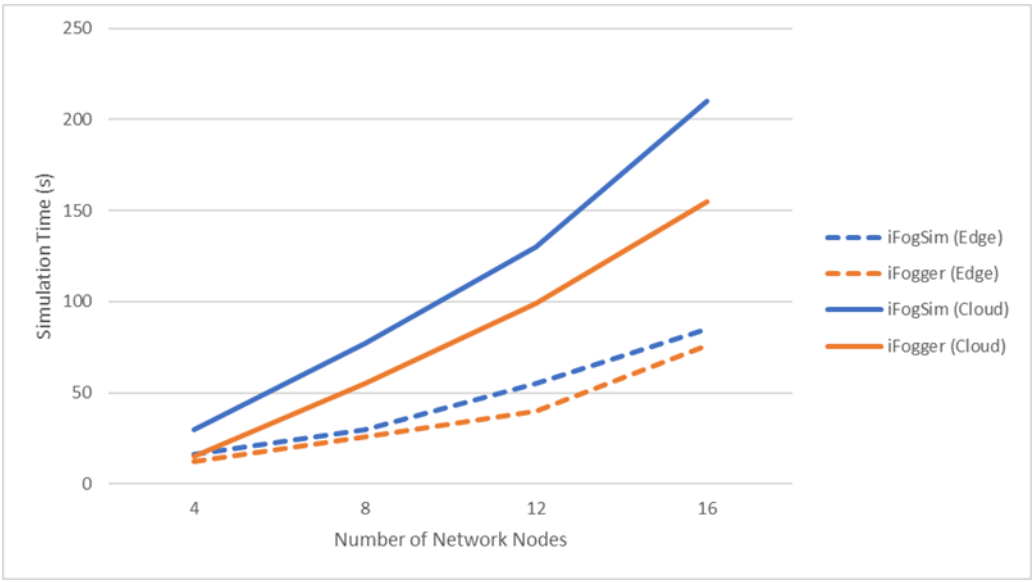

Figure 5. Simulation time results for the VR game example.

Both simulators exhibit similar behavior, but some differences can be observed:
I) Cloud policy requires more transmissions since all messages go through multiple network links to reach the cloud entity. This high volume of traffic may lead to network saturation and affect iFogSim's runtime.
II) Edge policy generates more application modules, leading to more events processes to control each module. This slightly affects iFogger runtime, which is reasonable due

to the increased number of modules, but the saturation of the simulated system should not affect the simulator itself.

5.Conclusions and Future Work

Fog computing is a computing paradigm extends cloud computing and bring the computing to the vacancy of users near computer edge devices. The cloud/edge/fog continuum serve IoT latency sensitive applications and real-time approaches. A Fog Computing simulator was proposed in this paper to serve deploying of resources placement algorithms named as iFogger. The simulator was adopted from iFogSim the widely used Fog simulator and built on top of OMNeT++ a commonly used network simulator. For future work the simulator, mobile nodes handover can be added as a new feature that iFogSim does not has. In addition to enhancing the messaging system to include network protocols such as MQTT and CoAP. The generated results can be extended to include network nodes queueing times to monitor the effects of placement algorithms on the behavior of these nodes.

6.References

[1] Silva, L., Magaia, N., Sousa, B., Kobusińska, A., Casimiro, A., Mavromoustakis, C. X., ... & De Albuquerque, V. H. C. (2021). Computing paradigms in emerging vehicular environments: A review. *IEEE/CAA Journal of Automatica Sinica*, *8*(3), 491-511.

[2] Das, R., & Inuwa, M. M. (2023). A review on fog computing: issues, characteristics, challenges, and potential applications. *Telematics and Informatics Reports*, 100049.

[3] Sabireen, H., & Neelanarayanan, V. J. I. E. (2021). A review on fog computing: Architecture, fog with IoT, algorithms and research challenges. *Ict Express*, *7*(2), 162-176.

[4] Angel, N. A., Ravindran, D., Vincent, P. D. R., Srinivasan, K., & Hu, Y. C. (2021). Recent advances in evolving computing paradigms: Cloud, edge, and fog technologies. *Sensors*, *22*(1), 196.

[5] Akana, C. M. V. S., Kumar, K. S., Divakar, C., & Satyanarayana, C. (2011). Dynamic resource allocation in computing clouds through distributed multiple criteria decision analysis using PROMETHEE method. *International Journal of Advanced Networking and Applications*, *3*(2), 1060.

[6] Hanumantharaju, R., Sowmya, B. J., Paul, A., Muralidhar, A., Aishwarya, R., Shriya, B. N., & Shreenath, K. N. (2022, November). Secured Fog-Based System for Smart Healthcare Application. In *Futuristic Trends in Networks and Computing Technologies: Select Proceedings of Fourth International Conference on FTNCT 2021* (pp. 185-197). Singapore: Springer Nature Singapore

[7] Chen, N., Yang, Y., Zhang, T., Zhou, M. T., Luo, X., & Zao, J. K. (2018). Fog as a service technology. *IEEE Communications Magazine*, *56*(11), 95-101.

[8] Das, R., & Inuwa, M. M. (2023). A review on fog computing: issues, characteristics, challenges, and potential applications. *Telematics and Informatics Reports*, 100049.

[9] Salaht, F. A., Desprez, F., & Lebre, A. (2020). An overview of service placement problem in fog and edge computing. *ACM Computing Surveys (CSUR)*, *53*(3), 1-35.

[10] Manishankar, S., Harshitha, S., Mukhopadhyay, A., & Anoop, A. (2019, March). Technologies for network testing: A hybrid approach. In *2019 3rd International Conference on Computing Methodologies and Communication (ICCMC)* (pp. 946-951). IEEE.

[11] Mahmud, R., Pallewatta, S., Goudarzi, M., & Buyya, R. (2022). Ifogsim2: An extended ifogsim simulator for mobility, clustering, and microservice management in edge and fog computing environments. *Journal of Systems and Software*, *190*, 111351.

[12] Khan, E. U. Y., Soomro, T. R., & Brohi, M. N. (2022, October). iFogSim: A Tool for Simulating Cloud and Fog Applications. In *2022 International Conference on Cyber Resilience (ICCR)* (pp. 01-05). IEEE.

[13] Qayyum, T., Malik, A. W., Khattak, M. A. K., Khalid, O., & Khan, S. U. (2018). FogNetSim++: A toolkit for modeling and simulation of distributed fog environment. *IEEE Access*, *6*, 63570-63583.

[14] Sonmez, C., Ozgovde, A., & Ersoy, C. (2018). Edgecloudsim: An environment for performance evaluation of edge computing systems. *Transactions on Emerging Telecommunications Technologies*, *29*(11), e3493.

[15] Lera, I., Guerrero, C., & Juiz, C. (2019). YAFS: A simulator for IoT scenarios in fog computing. *IEEE Access*, *7*, 91745-91758

[16] Lopes, M. M., Higashino, W. A., Capretz, M. A., & Bittencourt, L. F. (2017, December). Myifogsim: A simulator for virtual machine migration in fog computing. In *Companion proceedings of the10th international conference on utility and cloud computing* (pp. 47-52).

[17] Puliafito, C., Gonçalves, D. M., Lopes, M. M., Martins, L. L., Madeira, E., Mingozzi, E., ... & Bittencourt, L. F. (2020). MobFogSim: Simulation of mobility and migration for fog computing. *Simulation Modelling Practice and Theory*, *101*, 102062.

[18] Amarasinghe, G., de Assuncao, M. D., Harwood, A., & Karunasekera, S. (2020). ECSNeT++: A simulator for distributed stream processing on edge and cloud environments. *Future Generation Computer Systems*, *111*, 401-418.

[19] Markus, A., Al-Haboobi, A., Kecskemeti, G., & Kertesz, A. (2023). Simulating IoT Workflows in DISSECT-CF-Fog. *Sensors*, *23*(3), 1294.

[20] Kecskemeti, G. (2015). DISSECT-CF: a simulator to foster energy-aware scheduling in infrastructure clouds. *Simulation Modelling Practice and Theory*, *58*, 188-218.

[21] El Idrissi, M., Elbeqqali, O., & RIFfi, J. (2019, October). A review on relationship between Iot–cloud computing–fog computing (Applications And Challenges). In *2019 third international conference on intelligent computing in data sciences (ICDS)* (pp. 1-7). IEEE.

[22] Asif-Ur-Rahman, M., Afsana, F., Mahmud, M., Kaiser, M. S., Ahmed, M. R., Kaiwartya, O., & James-Taylor, A. (2018). Toward a heterogeneous mist, fog, and cloud-based framework for the internet of healthcare things. *IEEE Internet of Things Journal*, *6*(3), 4049-4062.

[23] Stojmenovic, I. (2014, November). Fog computing: A cloud to the ground support for smart things and machine-to-machine networks. In *2014 Australasian telecommunication networks and applications conference (ATNAC)* (pp. 117-122). IEEE.

[24] Lan, D., Liu, Y., Taherkordi, A., Eliassen, F., Delbruel, S., & Lei, L. (2021, March). A federated fog-cloud framework for data processing and orchestration: a case study in smart cities. In *Proceedings of the 36th annual ACM symposium on applied computing* (pp. 729-736).

[25] Varshney, S., & Singh, S. (2018). A survey on resource scheduling algorithms in cloud computing. *International Journal of Applied Engineering Research*, *13*(9), 6839-6845.

[26] Alqahtani, A. M., Yosuf, B., Mohamed, S. H., El-Gorashi, T. E., & Elmirghani, J. M. (2022). Energy Efficient VM Placement in a Heterogeneous Fog Computing Architecture. *arXiv preprint arXiv:2203.14178*.

[27] Tsai, J. F., Huang, C. H., & Lin, M. H. (2021). An optimal task assignment strategy in cloud-fog computing environment. *Applied Sciences*, *11*(4), 1909.

[28] Margariti, S. V., Dimakopoulos, V. V., & Tsoumanis, G. (2020). Modeling and simulation tools for fog computing—a comprehensive survey from a cost perspective. *Future Internet*, *12*(5), 89.

[29] Kurdi, H., & Thayananthan, V. (2022). A Multi-Tier MQTT architecture with multiple brokers based on fog computing for securing industrial IoT. *Applied Sciences*, *12*(14), 7173.

[30] Dizdarević, J., Carpio, F., Jukan, A., & Masip-Bruin, X. (2019). A survey of communication protocols for internet of things and related challenges of fog and cloud computing integration. *ACM Computing Surveys (CSUR)*, *51*(6), 1-29.

[31] Malik, A. W., Qayyum, T., Rahman, A. U., Khan, M. A., Khalid, O., & Khan, S. U. (2020). XFogSim: A distributed fog resource management framework for sustainable IoT services. *IEEE Transactions on Sustainable Computing*, *6*(4), 691-702.

[32] Chang, V., Sidhu, J., Singh, S., & Sandhu, R. (2023). SLA-based Multi-dimensional Trust Model for Fog Computing Environments. *Journal of Grid Computing*, *21*(1), 4.

[33] Singh, P., Kaur, R., Rashid, J., Juneja, S., Dhiman, G., Kim, J., & Ouaissa, M. (2022). A fog-cluster based load-balancing technique. *Sustainability*, *14*(13), 7961.

[34] Obelovska, K., & Danych, I. (2022, February). Adoption of the OMNET++ simulator for the computer networks learning: a case study in CSMA schemes. In *The International Conference on Artificial Intelligence and Logistics Engineering* (pp. 234-243). Cham: Springer International Publishing.