

Method of optimizing delay in IoT system using fog calculations

Sergii Lysenko^{1,†}, Oleh Bondaruk^{1,*,†}, Piotr Gaj^{2,†}, Anatoliy Sachenko^{3,4,†} and Olena Lysenko^{1,†}

¹ Khmelnytskyi National University, Khmelnytsky, Instytutska street 11, 29016, Ukraine

² Silesian University of Technology, ul. Akademicka 2A, 44-100 Gliwice, Poland

³ Kazimierz Pulaski University of Technology and Humanities, Department of Informatics, Radom, Poland

⁴ Research Institute for Intelligent Computer Systems, West Ukrainian National University, Ternopil, Ukraine

Abstract

The article presents the improved the method and means of task planning in IoT infrastructure using fog computing. For the purpose of improvement, the task planning method based on the ant colony algorithm (ACO) was chosen. The concept of priority of task execution by fog nodes was introduced. Formulas were also developed to determine the node that will perform calculations with the least time and delay. The environment for conducting DISSECT-CF experimental studies was also considered. Its modules, abstractions and their settings were described. According to the results of experimental studies, it was determined that the use of an improved method of optimizing the IoT infrastructure with the use of fog computing improved the quality of service (QoS) of the system by reducing the delay of processing tasks.

Keywords

delay optimizing, IoT systems, fog calculations

1. Introduction

Fog computing is a system-level architecture that helps achieve optimal distribution of network, computing power, and information storage. Fog computing combines the advantages of cloud and edge computing to provide high quality of service, reduce latency, ensure mobility and other functions used in modern computing systems. The application of fog computing provides a higher speed of data processing, since the computing power is closer to the IoT devices at the geographical level [1][2].


The purpose of using the concept of fog computing is to achieve a proper balance between the main characteristics and optimally organize the network [3]. Fog computing expands the possibilities of cloud computing due to the transfer of calculations to peripheral devices or

ICyberPhyS-2024: 1st International Workshop on Intelligent & CyberPhysical Systems, June 28, 2024, Khmelnytskyi, Ukraine

* Corresponding author.

† These authors contributed equally.

✉ sirogyk@ukr.net (S. Lysenko); oleg6467@ukr.net (O. Bondaruk); piotr.gaj@polsl.pl (P. Gaj); as@wunu.edu.ua (A. Sachenko)

 0000-0001-7243-8747 (S. Lysenko); 0009-0000-86634124 (O. Bondaruk); 0000-0002-2291-7341 (P. Gaj); 0000-0002-0907-3682 (A. Sachenko); 0009-0001-7169-5650 (O. Lysenko)



© 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

Internet of Things devices for cybersecurity [4][5]. In addition, fog computing extends cloud technologies to manage virtualization and edge-side security with the help of fog computing layers [6][7].

Using the concept of fog computing in the construction of IoT infrastructure improves speed, scalability, efficiency and security for important applications in IoT infrastructure [8]. Tasks of fuzzy computing: calculations, data storage, provision of network services on peripheral devices on the user side of the infrastructure [9].

The fog computing paradigm provides significant latency reduction and greater awareness system nodes in the context of task execution and supports vertically isolated applications sensitive to delays using continuous network connectivity and scaling [10].

The characteristics that include fog computing are:

- low latency - due to the close location to the end devices, a high speed of response and data processing is achieved [11];
- rich and diverse support for end devices - achieved due to the close location of computing nodes to end devices [12];
- multi-user controlled environment - thanks to a highly virtualized distributed platform;
- support for mobility - thanks to the direct communication of the application in a foggy environment with mobile devices [13];
- contextuality – devices and fog nodes have all the information about the environment in which they are located [14];
- geographical distribution – due to distribution, the fog environment ensures high quality of streaming services [15];
- wireless access - suitable for wireless sensor devices that require time-distributed analysis and communication;
- support for heterogeneity – fog environment nodes can have different form factors and be deployed in different distributed environments [16];
- device compatibility – functional compatibility with devices from different manufacturers in different industries [17];
- real-time analytics – available due to the close location of fog nodes to IoT devices;
- industrial use – has a wide range of industrial applications due to real-time data processing [18].

2. Related works

The algorithms proposed in [19][20][21] use the deployment of virtual machines for virtual medical devices and provides the necessary QoS of medical applications in real time, taking into account the cost of communication, the cost of calculations, the placement of virtual machines and the price of task distribution. The disadvantage of the proposed algorithms are that the proposed fog computing network architectures are based on a cellular network, in which fog nodes are represented as cellular base stations. In addition, the possibility of actual task scheduling and load balancing is not taken into account, but only the placement of virtual machines. As a result, the method cannot be generalized to the fog computing architecture.

The studies [22][23][24] considers the location problem of edge servers using a PCO optimization method based on a multi-objective function in order to reduce the total energy consumption and maintain a satisfactory delay.

Also, genetic algorithms are used to make decisions about offloading tasks in fuzzy computing. However, most studies do not consider the issue of load balancing between fog nodes.

The IoT fog network architecture consists of three layers: the IoT layer, the fog layer, and the cloud layer, as shown in Figure 6. The same architecture is used in the study [25][26][27]. In the IoT layer, a number of geographically distributed sensor nodes are connected using a local area network [28]. The IoT layer collects different data from different applications, such as weather, air quality, and traffic. The IoT layer is connected to the fog computing layer, which contains a number of fog nodes. Each fog node is used to aggregate, filter and process the collected data from the sensors. Each fog node contains a local agent that is responsible for collecting performance data such as sensor data arrival rate and sensor service rate.

Sensors send offload requests to fog nodes, and fog nodes forward these requests to the fog master node, which is responsible for scheduling offloading tasks to fog nodes using aggregated data from fog agents. The cloud tier provides powerful computing and storage capabilities. The fog layer is connected to the cloud layer through the Internet, not through a local network. The cloud level is used for intensive data processing and storage.

3. Method of optimizing delay in IoT system using fog calculations

In order to optimize the delay in the IoT system using fog computing, the task scheduling method based on the ACO algorithm was improved by implementing the execution priority for different classes of tasks in the system.

3.1. Priority of execution of planned tasks

IoT infrastructure using the concept of fog computing were determined:

1. Definition of classes of tasks that can be processed by nodes.
2. Allocation of priorities between classes of tasks within each of the fog nodes.
3. In determining the response delay of nodes using the ACO algorithm.
4. Adjusting the results of determining the efficiency of nodes using the priorities of task classes and the current load of fog nodes.

In the work, the ACO algorithm was used, which showed the best results in optimizing the reduction of response time when planning tasks. In order to reduce the average processing time of tasks, it was decided to add a priority condition for tasks on fog nodes. Thus, the fog node will focus computing power on certain classes of tasks and, accordingly, process a larger number of them per time unit. Figure 1 shows a network of fog nodes connected by task class. Task classes are marked with geometric figures. In this way, optimization will take place by class of tasks. It follows that instead of the total number of fuzzy nodes, the search for the fastest one will take place among the set of nodes performing a specific class or classes of tasks.

A suitable heterogeneous environment requires that gateways with load balancers must be present in the architecture. Such gateways will act as task schedulers. Since each gateway will have information about each fog node, its task classes, workload and response time, it can effectively plan by distributing tasks.

Figure 2 shows an architecture in which gateways act as task schedulers. They receive tasks from IoT devices and schedule the task, depending on its class, using the fog network response delay network table.

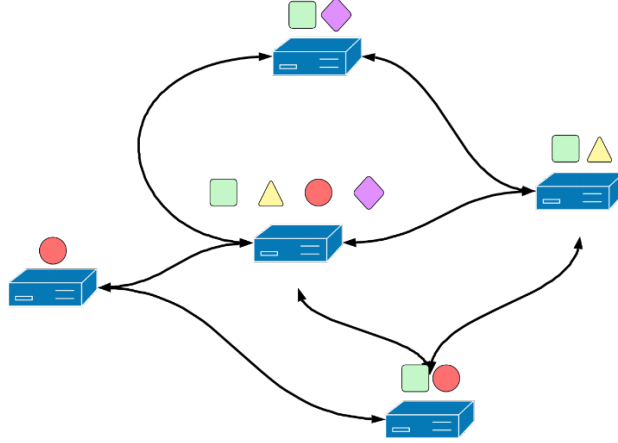


Figure 1: Connection of fog nodes by task classes of gateways and IoT devices in the system.

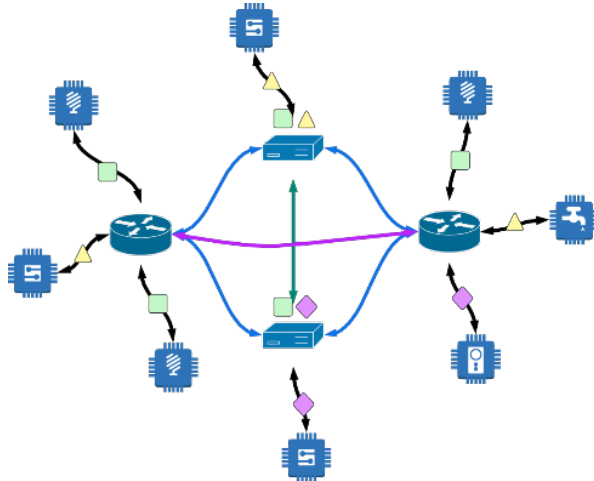


Figure 2: Architecture of gateways and IoT devices in the system.

From Table 1, N_i is a fog node, S_j is a task class, and L_{ij} – response delay of node N_i task of class S_j . It can be seen that node 1 and node 2 perform tasks C and D. However, node 1 should have a higher priority to perform task D, and node 2 should have a higher priority to perform task C. Accordingly, we have formula 1:

$$N_{min} = \text{Min}(L(N_i, S_j)), \quad (1)$$

where N_{min} is the node with the smallest response delay.

Sensors can generate a large amount of data at a high frequency, so there may be cases where all the fog nodes and the cloud environment are overloaded. In this case, the data received from

the sensors may be lost. Often such data loss may not be critical, but there are areas of IoT use where low latency is a major system requirement. For such a case, it is proposed to introduce the priority of task classes.

Table 1
Network table of response latencies

N_i	S_j	L_{ij}
1	A	12
1	B	34
1	C	78
1	D	40
2	C	37
2	D	140
3	E	22

Consider a node that processes tasks of class A and B. In the event that the execution of a task of class A takes 40 ms, and the task of class B takes 500 ms, during the execution time of a task of class B, 12.5 tasks of class A can be executed. In this case, you can impose the requirement of the lowest possible delay of class A tasks and the low importance of performing class B tasks. The implementation of such behavior requires a heuristic that will determine that a certain class of tasks has priority over others in case of task overload.

Such a heuristic can be achieved by implementing a task priority table.

Table 2
Task priorities

N_i	S_j	P_s
1	A	50
1	B	25
1	C	10
1	D	15
2	C	80
2	D	20
3	E	100

From the data in table 2, we can derive formula (2), which calculates the sum of priorities.

$$\sum P(N_i, S_j) = 100\%. \quad (2)$$

From the data in formula 2, it is possible to introduce the concept of a cycle of priorities. The priority cycle is a task planning cycle that will satisfy formula 2, i.e., 100% of tasks according to the planned priorities will be executed in one execution of the cycle.

As soon as the quotas of all task classes are exhausted, the counters will be updated and the next cycle of task processing will begin. If only tasks of one class arrive at the node, the

scheduler will continue to process them, updating the counter, until new tasks of other classes appear. Counters are used to balance tasks, otherwise tasks generated with a high frequency may overload the fog node, which will delay task execution. This behavior can lead to the loss of relevance of data that was processed with a delay. In order to determine by the scheduler, the node with the smallest delay, it is necessary to keep a counter of the classes of tasks that were processed by the node. In addition, it is necessary to enter a counter of receiving tasks by the node to save information not only about the load of the node by class of tasks, but also the total load of the node with tasks.

In order to determine the total load of the node with tasks, we will introduce formula 3, which will take into account both the number of completed tasks and the complexity of their processing:

$$n(P_i) = \sum \frac{\sum P(N_i, S_j) [t(S_j) \geq t_{current} - t(n)]}{t(n) \times L_{ij}} \quad (3)$$

where $n(P_i)$ is the total load of the node, n = the time window in which the calculations are performed, which is necessary in order to keep in memory only the relevant data on the processing of tasks, $t(S_j)$ is the time when the task was completed.

Formula 3 calculates the node load, which does not depend on the task class and takes into account the task execution delay. From here will receive a new network table 3, which will display complete information about the state of operation of the fog environment node.

Table 3
Network priority table with total load of nodes

N_i	S_j	L_{ij}	P_s	$P(N_i, S_j)$	$n(P_i)$
1	A	12	50	11	23.03
1	B	34	25	3	
1	C	78	10	6	
1	D	40	15	12	
2	C	37	80	40	36.4
2	D	140	20	7	
3	E	22	100	98	74

Using table 3, the task scheduler will be able to take into account the total load nodes.

The values in table 3 were calculated using formula 3, where the value of n is 60 seconds and the values of the counters $P(N_i, S_j)$ were generated within the time window n . Hence, according to formula 3, the values $n(P_i)$ show the total load of the node N_i .

It should be noted that for formula 3, the value of $n(P_i)$ will approach 1 when the use of node resources is close to the maximum. Also, when the values of $n(P_i)$ will exceed 1, it will mean that the node has a high computing power. If the value is significantly less than 1, it means that the node is free to load new tasks.

3.2. An environment for conducting experimental research on the operation of IoT infrastructure using fog computing

The DISSECT-CF- Fog simulation environment of IoT systems was used to conduct experimental studies of IoT infrastructure. DISSECT-CF- Fog is an open-source software environment. The environment offers the functionality of detailed modeling of the system in which it is possible to manage various parameters of the system, which allows you to set up an environment close to the real world.

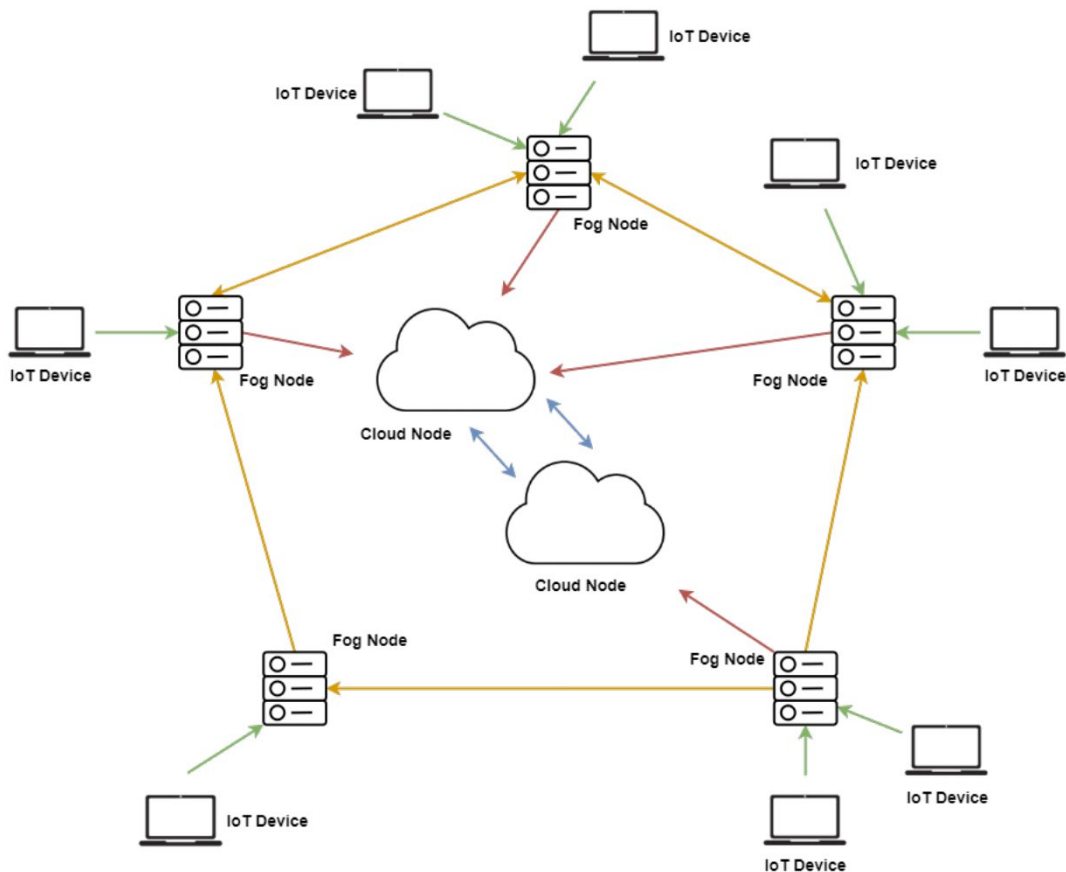


Figure 3: IoT network architecture using fog computing built in the DISSECT-CF- Fog environment [25]

The simulation tool provides the possibility to use different task scheduling algorithms, taking into account the energy usage, the geographical location of the IoT environment devices and the cost of the resources of the cloud environment, which is close to real cloud services. DISSECT-CF is a Discrete Event Simulation (DES) framework, so it supports decision making for complex time-related scenarios. The software is capable of simulating the internal processes of distributed systems, supporting parallel and distributed computing despite sequential execution. The DISSECT-CF- Fog extension used for fog network modeling consists of the

following main components: application is a component that provides a task scheduling mechanism that takes into account the load on nodes and distributes tasks between them. The characteristics of the component are Frequency, Task Size, Instance, Instruction Count, Threshold, Application Strategy, IoT Price. Parameters of the microcontroller component are CPU, Memory, Processing Power, Repository, Turned on Process, Turned off Process, Minimum Power, Idle Power, Maximum Power.

4. Experiments

In order to implement tools and improve the method of task planning in IoT infrastructure with the use of fog computing, the DISSECT-CF simulation environment was used, which provides the functionality of building an IoT system using fog computing.

The DISSECT-CF simulation tool allows you to collect information about the energy use of fog network nodes and cloud environments. In addition, it is possible to obtain information about the load of fog network nodes at different times. Five nodes of the fog environment and one instance deployed in the cloud were used for the experimental study. The location of nodes is shown in Figure 4. Also, the figure shows the device routes. Device routes were built based on the principle that one device should be covered by several fog nodes. A route was built in which, in certain areas, the device will not be covered by any of the fog nodes. Table 4 shows the parameters of fog network nodes. Table 5 contains a description of the parameters of the cloud environment.

Table 4

Fog network node settings parameters

Parameter name	Value
Random Access Memory	4 GB
Central processor	1 core
Computing power	0.001 instructions / ms
Startup process	100 instructions
solid state recorder	1 GB

Table 5

Cloud environment settings parameters

Parameter name	Value
Random Access Memory	8 GB
Central processor	2 cores
Computing power	0.002 instructions / ms
Startup process	100 instructions
Solid state drive	50 GB

Next, it is necessary to configure the parameters of the software environment that will process the tasks.

The parameters of the software environment deployed on the fog network nodes are: task size - the task size attribute indicates the maximum amount of raw data that can be packed into one computing task for execution by virtual machines.

The simulation environment guidelines recommend parameter input values greater than 5000 bytes; frequency - Based on the frequency value, the daemon service checks the storage for raw data. According to the instructions of the simulation environment, the value should be greater than 6000 milliseconds; the number of instructions is a parameter that determines the maximum value that can be represented by one task. The recommended value is greater than 1000; allocation - determines how many unprocessed tasks can be held in a real application, subsequent tasks will be forwarded according to one of the application's strategies. Recommended value is 1-5.

Using the DISSECT-CF- Fog modeling functionality, the visualization of the network on the map, shown in Figure 4, was generated. The simulation environment has a set of built-in task allocation strategies in case the number of unprocessed tasks exceeds the data transfer allocation value: Random - chooses node fog random way; Push Up – always selects the parent node to which the node is attached. Hold Down - sends the task to the node closest to the end user. Runtime – selects the node with the lowest response latency, available CPU resources, and overall CPU characteristics. Pliant is a strategy that chooses a node based on the node load and the price of data processing. Therefore, it is suitable for modification and use in conjunction with the priority queue. In this way, the algorithm for determining the node for solving the problem will look like this:

- determination of node response delays using the ant colony algorithm (ACO);
- determination of available resources, taking into account the energy profiles of nodes using the swarm algorithm (PSO) ;
- determination of node load according to formula 3;
- definition of the task that will be performed according to the developed task planning algorithm.

To test the load of the network of fog nodes, 200 end devices continuously moving along routes and sending data from sensors were used. Table 6 shows the main characteristics of end devices.

Table 6
Parameters of the software environment

Parameter name	Value
The size of the task	50000 bytes
Frequency	60000 ms
Number of instructions	1000
Distribution of data transmission	2
Parameter name	Value

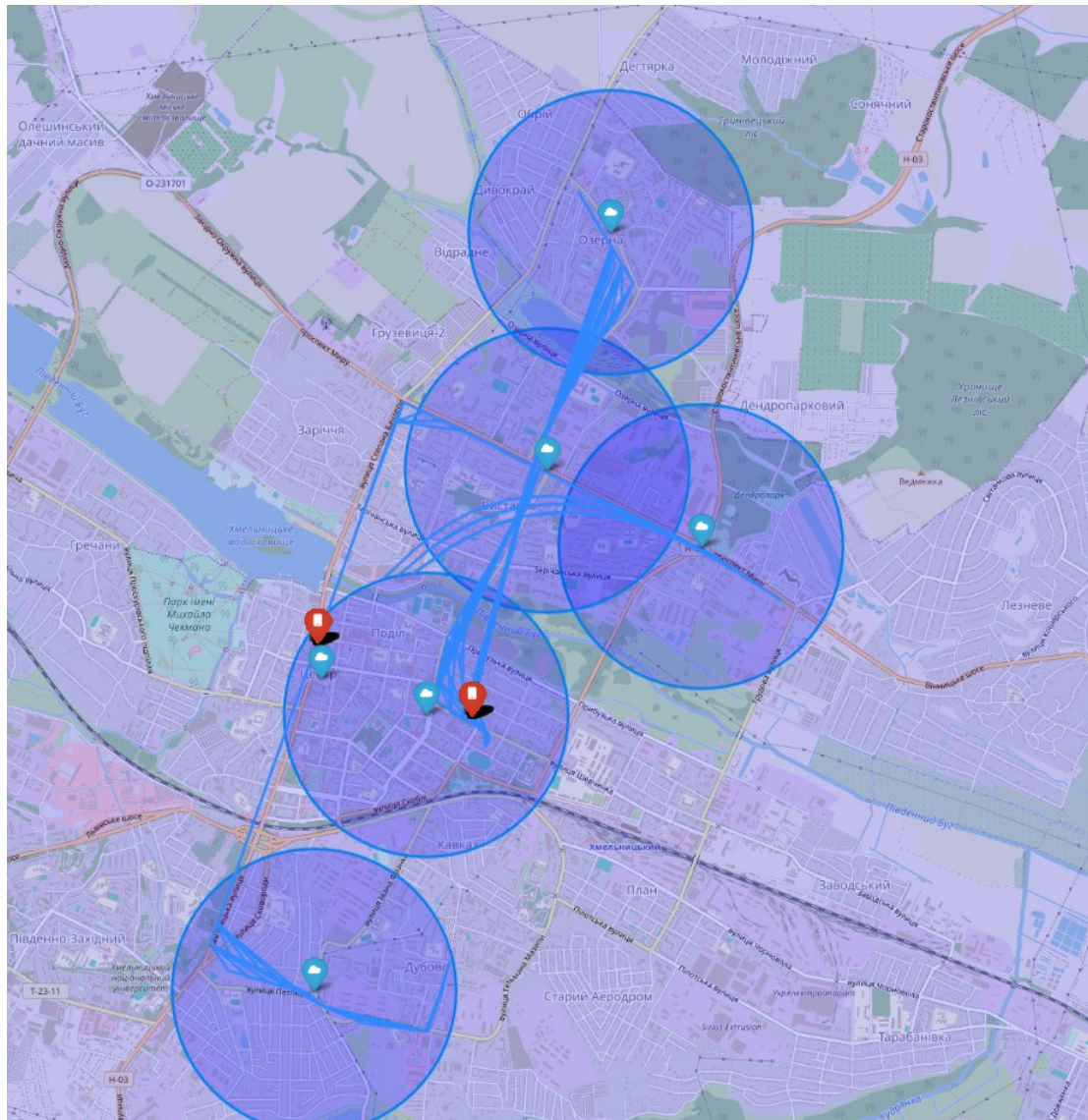


Figure 4: Location of fog network nodes and device routes

Fog simulation environment has a built-in function of generating graphs from data on system functioning: use; the cost of computing power on popular cloud environments; use of RAM; the number of completed and planned tasks; use of working hours of node processors; number processed data.

To compare the results, two models were used, one of which is based on the developed method of planning tasks in the IoT infrastructure using fog computing, the second is based on the search for the nearest device using the Hold Down planning strategy - a built-in planning strategy in the DISSECT-CF- Fog environment.

Figure 5 shows a graph of task planning comparison. The task scheduling graph shows the number of scheduled tasks in certain time intervals.

The blue curve showing the number of planned tasks for the developed planning strategy has a gap in the number of planned tasks by the Hold Down strategy. This result is caused by the details of the work of the developed algorithm based on task priorities. On an extended scale, it will be seen that this behavior is cyclical. The blue scale will move away and approach the orange scale. The algorithm schedules tasks in the order of arrival, but when the maximum value of the counter for the task class is reached, the algorithm starts to schedule tasks with higher priority.

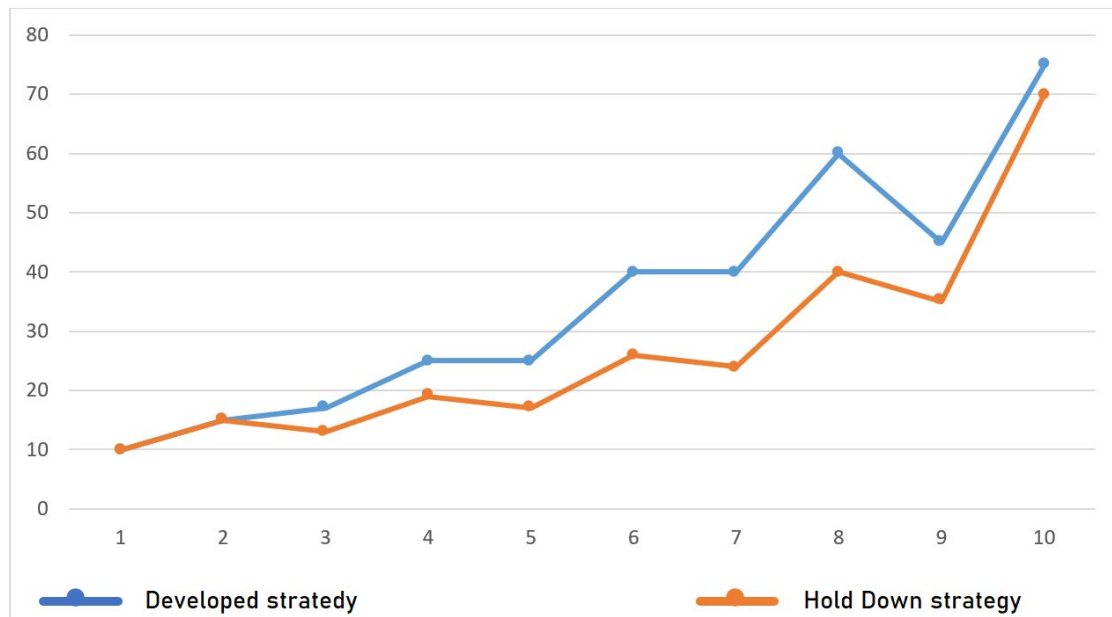


Figure 5: Task planning schedule.

In the case of this model, priority is given to tasks with a shorter processing time, so the scale for the developed strategy shows a larger number of planned tasks, but in the case that priority is given to tasks with a longer processing time, the result will be the opposite.

As a conclusion, it can be noted that a better result will be achieved if tasks that take less time to complete have a higher priority.

Based on the results of the experiment, a schedule of energy consumption by central processors of fog nodes was drawn. Figure 6 shows the obtained values of the total energy consumption of fog node processors according to the developed planning strategy and the Hold Down strategy.

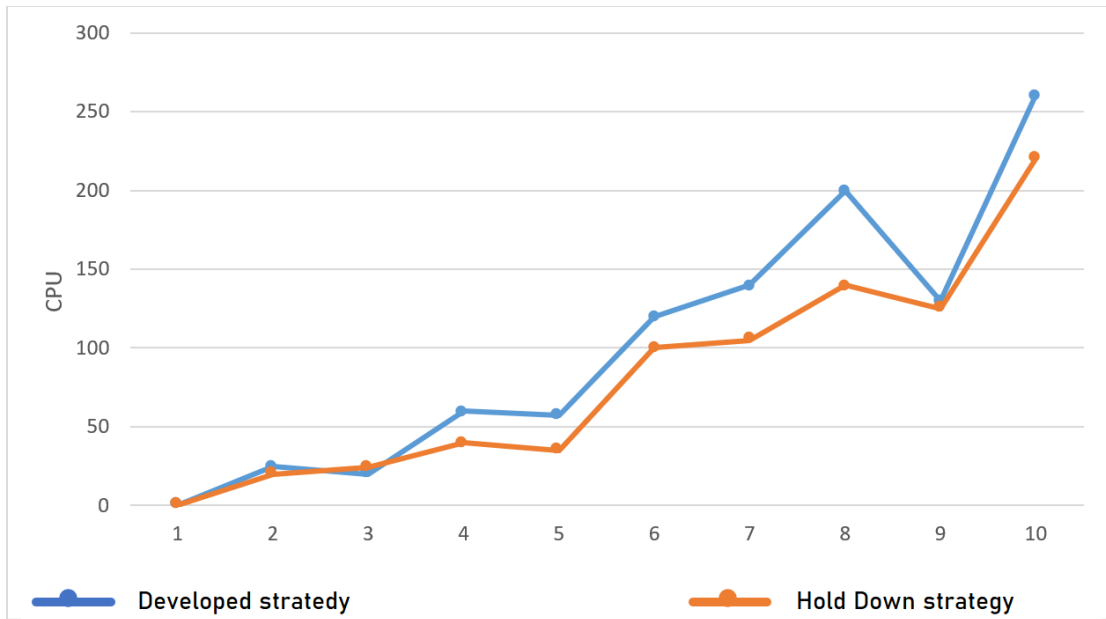


Figure 6: Total power consumption of fog node processors, in Watt.

Figure 6 fully reflects the behavior of the developed planning strategy, in which the iterative process of the priority task planning algorithm begins.

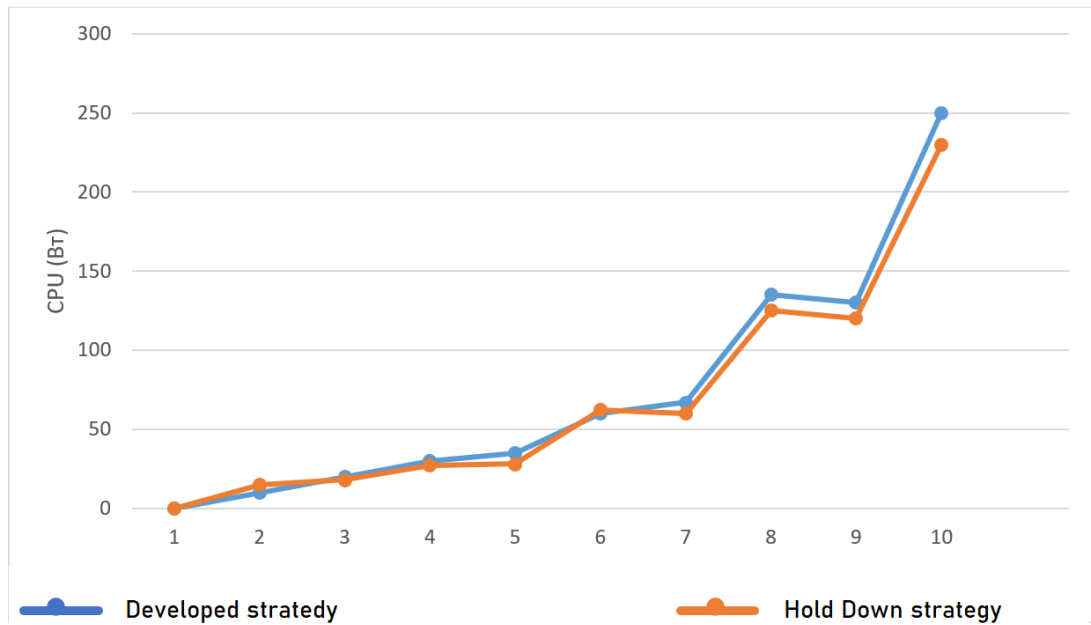


Figure 7: Energy consumption with priority on heavy tasks power consumption

Similarly to Figure 4, the values of energy consumption by node processors for the developed planning strategy are greater than the Hold Down strategy. The obtained indicator is the result of the fact that the algorithm began to perform more tasks with high priority and low latency. Unlike the Task Scheduling Schedule, if tasks with a longer execution time will be prioritized, the results of the energy consumption value using the developed scheduling strategy acquire close values to the Hold Down strategy, which is shown in Figure 7.

A graph of RAM consumption by fog nodes was also generated. The values on the graph show that the nodes in which the movement routes of end devices intersect are more loaded than the rest of the system nodes. RAM usage statistics are shown in Figure 8.

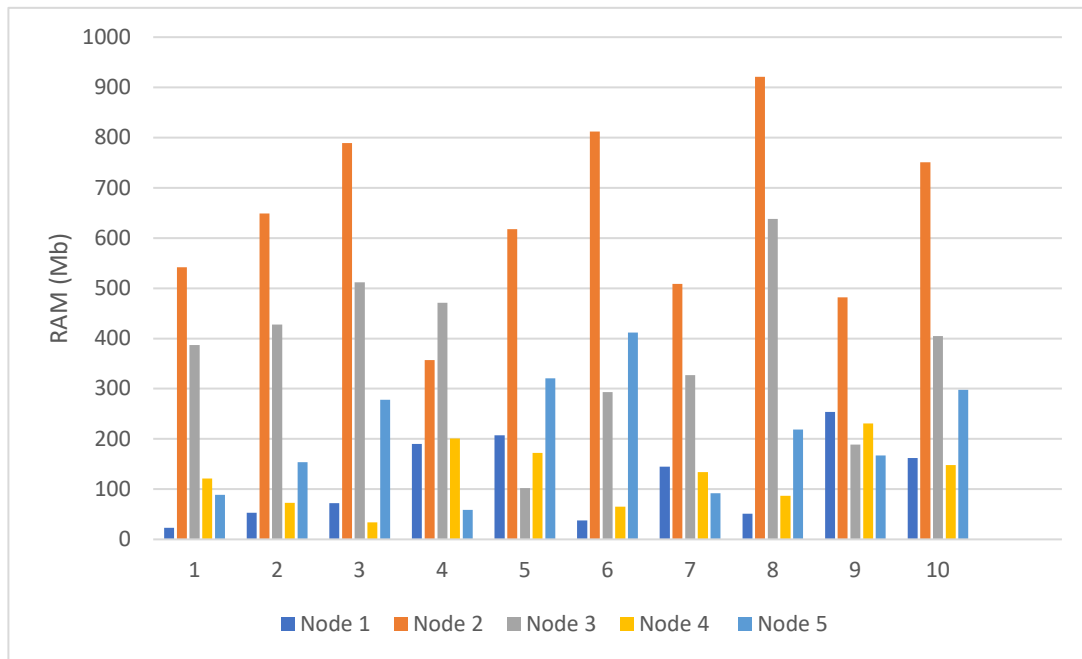


Figure 8: Total RAM consumption by fog nodes.

In addition to the fog nodes, computations were also performed on the cloud side, where all computation results from the fog nodes were sent, and computations in the case where the end device was not covered by the fog network or when the cloud environment had a lower response latency than the fog nodes. Figure 9 shows a graph of RAM usage on the cloud side.

The main parameter of optimization by the method that was developed is the delay of the response to the processing of the task by fog nodes. Response latency statistics were generated for all fog and cloud environment nodes.

Two classes of problems were used for the experiment: task class A has a low byte size, so it is quickly delivered by the network and processed by node. the number of completed and planned tasks; task class B has a large number of bytes and takes a long time to be downloaded by the network and takes more time to be processed by a node.

The average value of the task execution delay for the developed optimization method is presented in Figure 10. The chart shows the delay values for task classes A and B.

Figure 11 shows the experimental results for the Hold Down scheduling strategy, which also shows the delay values for problem classes A and B. Let's compare the delay for both strategies. Figure 12 shows a direct comparison of the average latency of class A tasks.

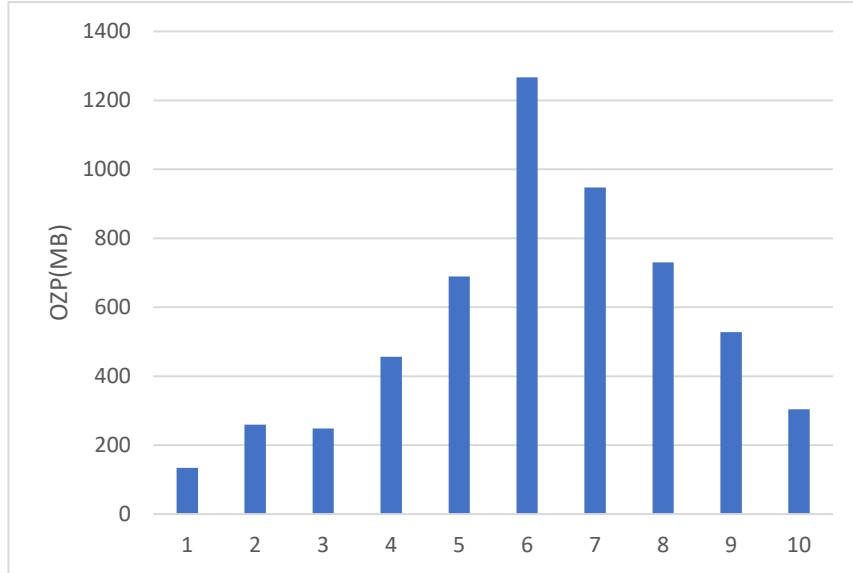


Figure 9: Use of RAM by the cloud environment.

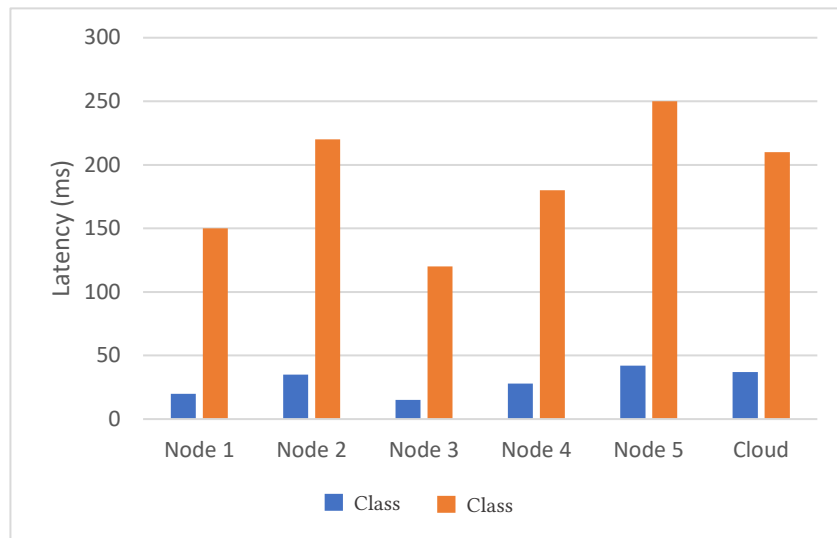


Figure 10: The average value of the response delay for processing tasks from nodes to end devices for the developed optimization method.

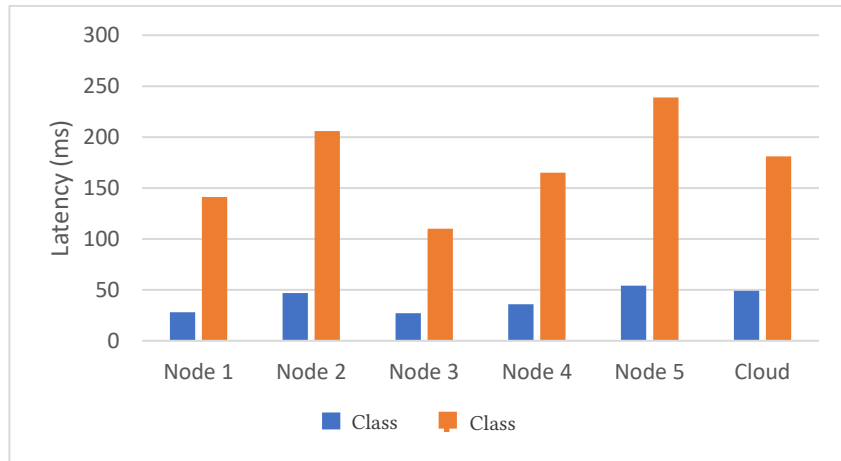


Figure 11: The average value of the response delay for processing tasks from nodes to end devices for the Hold Down strategy.

Figure 12 shows that the developed task planning strategy shows a better result of the delay in the execution of class A tasks. This result is due to the fact that planning takes place according to the planning algorithm based on the priority of tasks.

Figure 13 shows the average value of the delay in the execution of class B tasks. As a result, we can observe that for the Hold Down strategy the value of the delay in the execution of class B tasks is smaller than in the developed strategy. This is due to the fact that most of the task execution time falls on tasks of class A, which has a higher execution priority.

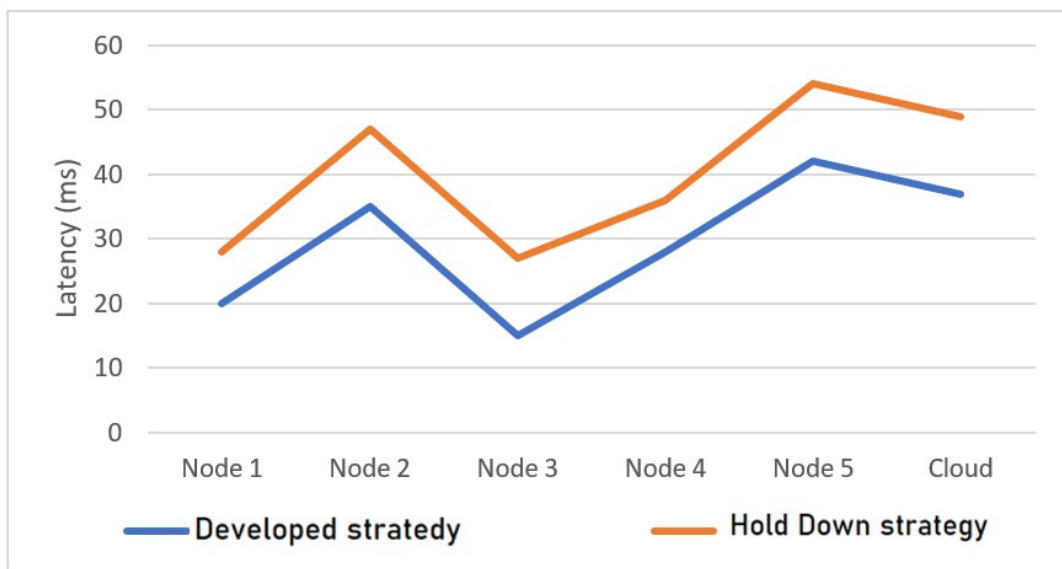


Figure 12: Comparison of the average value of the delay in the execution of class A tasks.

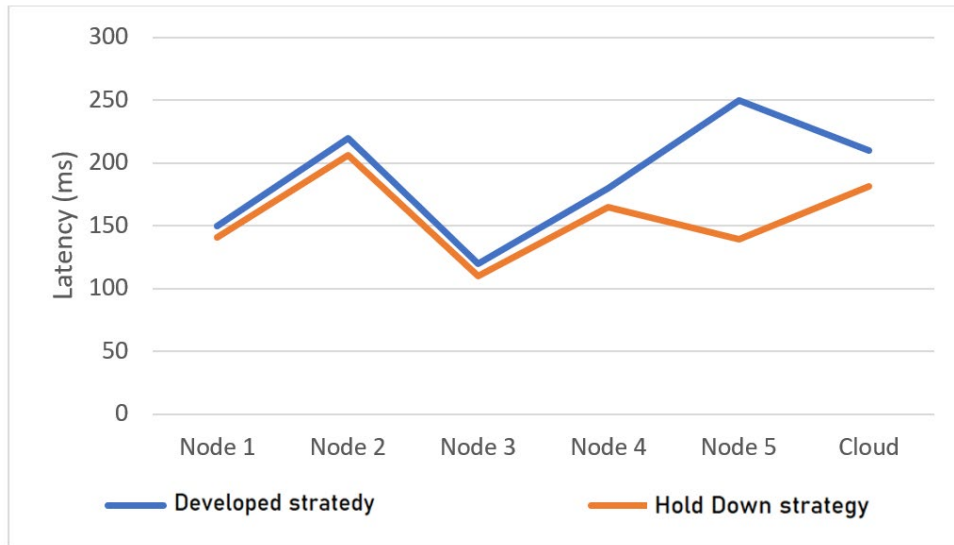


Figure 13: Comparison of the average value of the delay in the execution of class B tasks.

As a conclusion based on the result of the experiment, it can be determined that increasing the priority of tasks with a shorter processing time leads to a more productive operation of the IoT system as a whole. By using the developed task scheduling strategy, tasks with higher priority are executed with lower delays. Also, more such tasks are performed than could be performed without the use of priorities. Although the result of the developed strategy does not have a large difference in values with the strategy that chooses the nearest node. It can be concluded that the developed strategy increases the efficiency of the IoT network in the event that the majority of generated tasks have a low processing time.

5. Conclusion

The article presents the improved the method and means of task planning in IoT infrastructure using fog computing. For the purpose of improvement, the task planning method based on the ant colony algorithm (ACO) was chosen. The concept of priority of task execution by fog nodes was introduced. Formulas were also developed to determine the node that will perform calculations with the least time and delay. The environment for conducting DISSECT-CF experimental studies was also considered. Its modules, abstractions and their settings were described. According to the results of experimental studies, it was determined that the use of an improved method of optimizing the IoT infrastructure with the use of fog computing improved the quality of service (QoS) of the system by reducing the delay of processing tasks.

A comparison was made of the improved method of optimizing the IoT infrastructure using fog computing with the task scheduling method built into the simulation environment to the end device of the fog network node. In the infrastructure that used an advanced IoT infrastructure optimization method using fog computing, the response time was reduced by 30% for a class of tasks with low processing time. However, the response time for tasks that require more processing time, was increased by 6%.

According to the results of experimental studies, it was concluded that the improved method of IoT infrastructure optimization using fog computing increases the efficiency of the infrastructure if the vast majority of tasks generated have low processing time.

References

- [1] S. N. Srirama, A decade of research in fog computing: Relevance, challenges, and future directions. *Software: Practice and Experience*, 54(1) (2024). 3-23.
- [2] R. Rawat, R. K. Chakrawarti, P. Vyas, J. L. A. Gonzáles, R. Sikarwar, R. Bhardwaj, Intelligent fog computing surveillance system for crime and vulnerability identification and tracing. *International Journal of Information Security and Privacy (IJISP)*, 17(1), (2023). 1-25.
- [3] O. Savenko, A. Sachenko, S. Lysenko, G. Markowsky, N. Vasylykiv, Botnet detection approach based on the distributed systems. *International Journal of Computing*, 19(2), (2020) 190-198. <https://doi.org/10.47839/ijc.19.2.1761>.
- [4] G. Markowsky, O. Savenko, S. Lysenko, A. Nicheporuk, The technique for metamorphic viruses' detection based on its obfuscation features analysis. *CEUR-WS*, 2018. Vol. 2104. Pp. 680–687.
- [5] S. Lysenko, O. Savenko, K. Bobrovnikova. DDoS Botnet Detection Technique Based on the Use of the Semi-Supervised Fuzzy c-Means Clustering / *CEUR-WS* (2018), vol.2104, pp. 688-695.
- [6] S. Lysenko, O. Pomorova, O. Savenko, A. Kryshchuk, K. Bobrovnikova, DNS-based Anti-evasion Technique for Botnets Detection. *Proceedings of the 8-th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, Warsaw (Poland), September 24–26, 2015. Warsaw, 2015. Pp. 453–458.*
- [7] H. K. Apat, R. Nayak, B. Sahoo, A comprehensive review on Internet of Things application placement in Fog computing environment, *Internet of Things*, Volume 23, 2023, 100866, <https://doi.org/10.1016/j.iot.2023.100866>.
- [8] S. Iftikhar, M. M. M. Ahmad, S. Tuli, D. Chowdhury, M. Xu, S. S. Gill, S. Uhlig, HunterPlus: AI based energy-efficient task scheduling for cloud–fog computing environments. *Internet of Things*, 21, (2023). 100667.
- [9] J. Kaur, A. Agrawal, R.A. Khan, Security issues in fog environment: a systematic literature review. *Int J Wirel Inf Netw.* 2020; 27: 467-483.
- [10] K. Prateek, F. Altaf, R. Amin, S. Maity, A privacy preserving authentication protocol using quantum computing for V2I authentication in vehicular ad hoc networks. *Secur Commun Netw.* 2022; 2022: 1-17.
- [11] K. Prateek, S. Maity, Post-quantum blockchain–enabled services in scalable smart cities. *Quantum Blockchain: An Emerging Cryptographic Paradigm.* New York: John Wiley & Sons, Ltd; 2022: 263-291.
- [12] K. Prateek, S. Maity, Amin R. An unconditionally secured privacy-preserving authentication scheme for smart metering infrastructure in smart grid. *IEEE Trans Netw Sci Eng.* 2022; 10: 1085-1095.
- [13] B. Varghese, N. Wang, D.S. Nikolopoulos, R. Buyya, Feasibility of fog computing. In: R Ranjan, K Mitra, P Prakash Jayaraman, L Wang, AY Zomaya, eds. *Handbook of Integration*

- of Cloud Computing, Cyber Physical Systems and Internet of Things. Cham: Springer; 2020: 127-146.
- [14] P. Habibi, M. Farhoudi, S. Kazemian, S. Khorsandi, A. Leon-Garcia, Fog computing: a comprehensive architectural survey. *IEEE Access*. 2020; 8: 69105-69133.
- [15] A.A. Laghari, K. Wu, R.A. Laghari, M. Ali, A.A. Khan, A review and state of art of Internet of Things (IoT). *Arch Comput Methods Eng*. 2021; 29: 1395-1413.
- [16] J. Bellendorf, Z.Á. Mann, Classification of optimization problems in fog computing. *Future Gener Comput Syst*. 2020; 107: 158-176.
- [17] R. Mahmud, K. Ramamohanarao, R. Buyya, Application management in fog computing environments: a taxonomy, review and future directions. *ACM Comput Surv*. 2020; 53(4): 1-43.
- [18] J. McChesney, N. Wang, A. Tanwer, E. De Lara, B. Varghese, Defog: fog computing benchmarks. *Proceedings of the Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*; 2019:47-58.
- [19] M. Waqas, K. Kumar, A. A. Laghari, et al. Botnet attack detection in Internet of Things devices over cloud environment via machine learning. *Concurr Comput*. 2022; 34(4):e6662.
- [20] T. Hewa, A. Braeken, M. Liyanage, M. Ylianttila, Fog computing and blockchain-based security service architecture for 5G industrial IoT-enabled cloud manufacturing. *IEEE Trans Industr Inform*. 2022; 18(10): 7174-7185.
- [21] R. Nazir, A.A. Laghari, K. Kumar, S. David, M. Ali, Survey on wireless network security. *Arch Comput Methods Eng*. 2021; 29: 1591-1610.
- [22] C. Gonçalves DM, Lopes MM, et al. MobFogSim: simulation of mobility and migration for fog computing. *Simul Model Pract Theory*. 2020; 101:102062.
- [23] I. Hafid AS, Jarray A. Design, resource management, and evaluation of fog computing systems: a survey. *IEEE Internet Things J*. 2021; 8(4): 2494-2516.
- [24] A. Yousefpour, G. Ishigaki and J. P. Jue, "Fog computing: Towards minimizing delay in the Internet of Things", *Proc. IEEE Int. Conf. Edge Comput. (EDGE)*, pp. 17-24, Jun. 2017.
- [25] L. Gu, D. Zeng, S. Guo, A. Barnawi and Y. Xiang, "Cost efficient resource management in fog computing supported medical cyber-physical system", *IEEE Trans. Emerg. Topics Comput.*, vol. 5, no. 1, pp. 108-119, Jan. 2017.
- [26] R. Deng, R. Lu, C. Lai, T. H. Luan and H. Liang, "Optimal workload allocation in fog-cloud computing towards balanced delay and power consumption", *IEEE Internet Things J.*, vol. 3, no. 6, pp. 1171-1181, Dec. 2016.
- [27] O. Morozova, A. Tetskyi, A. Nicheporuk, D. Kruvak, V. Tkachov, Smart home system security risk assessment. *Computer Systems and Information Technologies*, (3), (2022). 81–88. <https://doi.org/10.31891/CSIT-2021-5-11>.
- [28] N. Harum, Z. Z. Abidin, W. M. Shah, A. Hassan, Implementation of smart monitoring system with fall detector for elderly using iot technology. *International Journal of Computing*, 17(4), 2018, 243-249. <https://doi.org/10.47839/ijc.17.4.1146>