Security and Performance Evaluation of Edge Computing Frameworks in Smart Environments

Donna Parekh[1]
Ph.D. Scholar, School of Information Technology, Indira University, Pune.

Dr. Madhavi Avhankar[2]
Ph.D. Guide, School of Information Technology, Indira University, Pune.

DOI: 10.29322/IJSRP.16.02.2026.p17025
https://dx.doi.org/10.29322/IJSRP.16.02.2026.p17025

Paper Received Date: 6th January 2026
Paper Acceptance Date: 7th February 2026
Paper Publication Date: 12th February 2026

Abstract

This paper provides a comprehensive evaluation of the security and performance aspects of edge computing frameworks in intelligent environments, including smart cities, residences, and industrial IoT systems. As more IoT devices come online, traditional cloud-based models have trouble meeting the needs of real-time programs that need low latency and high bandwidth. Edge computing solves these problems by processing data closer to where it originates from. This improves system responsiveness and scalability. On the other hand, moving to decentralized systems, it raises difficult security issues, like larger attack surfaces and worries about privacy. It also affects performance measures like latency, throughput, and how much energy it uses. This study conducts a comprehensive literature review to examine prevalent edge architectures, identify critical security vulnerabilities, and assess existing mitigation strategies. It also shows that how to monitor performance and how to find the correct balance between making the system safer and keeping it running at its best. The results show that we need solutions that are both flexible and work together to make sure that smart environments with edge computing are safe and work well.

Introduction

The Internet of Things (IoT) has led to a huge increase in the number of linked devices, which has revolutionized the way we monitor and manage modern surroundings [1], [2]. Smart environments, such smart cities, smart homes, and industrial IoT devices, are always sending massive amounts of data to the network edge. Centralized data centers are what traditional cloud computing is based on. Centralized data centers can only run a few programs since they need real-time processing, minimal latency, and efficient use of bandwidth. This is because data has to physically move from its source to cloud servers that are far away, which causes delays and more network traffic. This means that centralized solutions aren't good for applications that are sensitive to latency, such as self-driving cars and monitoring vital infrastructure [3], [4].

Edge computing has emerged as a distributed computing paradigm. This is meant to get around the problems by moving processing, storage, and networking closer to where the data is created. Processing data locally lowers end-to-end latency and makes backhaul networks less busy, which makes IoT apps more responsive and reliable. These properties make edge computing extremely useful for smart environments that need quick decisions and constant input. These environments are the main ones that benefit from edge computing's strengths.

Even if these are good things, switching from centralized cloud models to distributed edge frameworks makes security and performance harder. Hierarchical cloud–edge–device architectures make systems more scalable and flexible when it comes to deployment, but they also make them more complicated [5]. Edge nodes are spread out across a vast area, and there are not many computing resources available. This makes the attack surface much larger than with standard cloud infrastructures [8]. Because of this, it is very important to make sure that data is safe, private, and available in all of these places. At the same time, it is important to keep performance high, taking into account things like latency, throughput, energy use, and resource use, in order to meet the operational needs of smart environment apps.

In edge computing, security and performance are strongly related. Putting in place strong security measures often means more work for computers and communication, which can slow things down and use more energy. On the other hand, putting performance optimization first could unintentionally make the system less secure. Consequently, the assessment of edge computing frameworks

necessitates a simultaneous consideration of security and performance, rather than treating them as distinct design objectives. This study looks at the architectural designs of edge computing frameworks used in smart environments, finds important security issues, looks into ways to reduce these dangers, and goes over typical ways to evaluate performance. The report also points up trade-offs, problems with integration, and new areas of research in this quickly changing sector.

This study utilizes a systematic literature analysis to evaluate the security and performance of edge computing frameworks inside smart environments. The process encompasses a comprehensive search, selection, and critical assessment of academic publications, including peer-reviewed journal articles, conference papers, and authoritative technical reports. The emphasis was on research conducted in the last decade to encapsulate the most recent advancements and obstacles in edge computing, particularly those directly associated with smart environment applications or pertinent foundational technologies. The first search utilized words like "edge computing," "smart environments," "IoT," "security," "performance evaluation," "latency," "resource management," "privacy," "frameworks," and "architectures." These were merged with Boolean operators and utilized across principal scientific databases and digital libraries. We looked at the titles and abstracts to get rid of studies that weren't useful, and then we read the complete text of papers that might be useful. Empirical studies, surveys, and theoretical work that provide real data, architectural designs, or comparative evaluations of security mechanisms and performance metrics in edge computing environments were prioritized.

The main goal of data extraction was to compile a list of: - Specific edge computing designs that were suggested or looked at.- Found security holes and ways to attack them.- Suggested security solutions and the basic ideas behind them.- Performance measures that are used to evaluate things like latency, throughput, energy efficiency, and resource use.- Used benchmarking tools, simulation platforms, or experimental settings.- Described case studies or application situations.

The data that was gathered was put through thematic analysis to uncover patterns, common problems, and common solutions. This made it possible to combine security and performance data from different studies in a systematic way. For instance, security holes were put into three groups: data integrity, confidentiality, and availability concerns. Performance evaluations were based on the measurements and methodologies used. The investigation also looked at how security and performance affect one other and gave examples of when one affects the other. This methodical methodology gives us a strong and complete base from which to make conclusions and find new study paths.

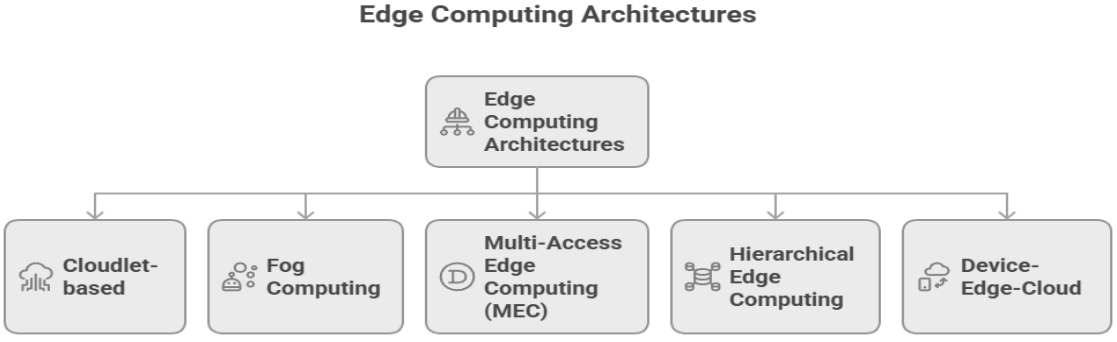
A look at the literature and a thematic analysis

The subsequent sections amalgamate insights from contemporary literature regarding edge computing architectures, security challenges, performance metrics, and privacy protocols in intelligent environments. This analysis points out the primary issues and the most common methods.

Smart environments and systems for edge computing

Edge computing is very different from centralized cloud infrastructures because it puts processing and data storage closer to where the data comes from. AI-driven intelligent edge computing makes it easier to make your own decisions [7].

This design is great for smart environments where IoT devices create a lot of data that needs to be handled immediately away. Different types of architecture have been created to fulfill the needs of different types of applications. One common way to do this is to have IoT devices connect to edge nodes, which then connect to a central cloud. This setup enables the edge handle data locally, which speeds things up and uses less bandwidth. It also sends tasks that aren't as vital or that need a lot of processing power to the cloud.



Mobile Edge Computing (MEC) is a special example of edge servers being located near cellular base stations or access points. These servers are mainly for mobile devices and users. This makes it possible to move computation-heavy and latency-sensitive apps off of mobile devices that don't have a lot of resources, like those in vehicular networks. For instance, vehicular edge computing can move jobs from nearby vehicles to other vehicles using algorithms that try to keep the average time latency as low as possible while also taking energy limits into account. When artificial intelligence (AI) is added to edge computing, it makes edge computing smarter. AI models make edge nodes smarter, but this also creates new security and privacy problems.

Another type of architecture is mist computing, which takes edge capabilities even further to very limited devices[6]. This type of computing focuses on collaboration and resource sharing between different types of devices. The Internet of Ships (IoS) is a real-world example of how edge computing may decentralize resources to support real-time communication, smart access, and privacy protection. This solves some of the problems with typical cloud-based IoS models. Cloud-edge-device paradigms are becoming the norm. They let IoT apps be deployed and scaled in a variety of ways, from device to cloud. Amazon Web Services (AWS) IoT Core and AWS Greengrass are two examples of platforms that provide cloud capabilities to edge devices. They enable local data processing and machine learning at the edge, which is especially useful in places with poor internet access.

Federated learning (FL) is becoming more common in edge architectures, especially in IoT settings like smart homes or self-driving cars. This way of learning together lets models be trained on decentralized edge devices without putting all the raw data in one place. This solves privacy issues, but it needs good key management and encryption systems. Digital Twins (DT) also use edge computing to help with resource scheduling in places like 5G-enabled distribution grids. A federated learning-based DT framework, for instance, can improve iteration delay and loss functions while also adding the ability to recognize unusual models for safe, fast, and accurate DT operations.

Security issues

Security issues are significantly harder to deal with in edge computing environments because they are spread out and different from each other. The system is more vulnerable to cyber attacks when there are many edge nodes in different parts of the world [8]. Edge devices are often limited in their resources, placed in easy-to-reach locations, and heavily reliant on wireless communication, which makes them even more susceptible to attacks [10]. Interoperability and heterogeneity remain substantial hurdles for deployment[20].

Data integrity, confidentiality, and availability across the device–edge–cloud continuum are essential security concerns in edge computing. Attackers have an easier time finding ways to break in since there are so many linked devices and different communication protocols. On the other hand, interoperability and heterogeneity make it tougher to apply security rules consistently across all deployments [20]. Physical tampering, illegal access, and compromised edge nodes are big problems, especially in applications where safety is very critical, including smart grids and self-driving cars [10].

More and more attacks are happening on the smart elements of edge computing systems. In intelligent edge environments, model poisoning, backdoor attacks, and adversarial manipulation of machine learning models can undermine security and functional accuracy [9]. These kinds of attacks not only make consumers less likely to trust edge-based intelligence, but they also slow down the system, which is bad for real-time applications that demand speedy inference.

You need a layered security system to fix these issues. You need fundamental technologies like secure communication protocols, encryption, and access control to keep data and devices safe at the edge.

How to Set Standards and See How Well People Are Doing

To find out how well edge computing frameworks work, especially in dynamic smart environments, performance evaluation is needed. The goal is to find out how well these systems can use all of their resources, analyze data in real time, and have low latency. Latency, throughput, energy use, resource use (CPU, memory, bandwidth), and job success rates are some of the most important performance metrics.

Latency [15] is one of the best methods to tell how well edge computing works. Latency is the time it takes for data to be made, processed, or responded. This is especially significant in smart environments where quick responses are needed. Edge computing seeks to cut this down a lot by processing data closer to where it came from. In other cases, like car networks, this might make things up to 68% better than running the program on your own PC. Throughput is another important number that tells you how much data is sent or processed in a certain length of time. This is really important for apps that work with a lot of data from many IoT devices.

For edge devices that run on batteries[17], energy utilization is particularly important. It also helps us figure out how long edge deployments can survive. By examining at resource consumption data, we can see how well job offloading and resource allocation methods work across edge nodes.Strategies for offloading jobs cut down on the time it takes to run a job by a lot [16].

You can also tell how strong and stable a system is by looking at how often tasks are completed successfully, especially when the connection is shaky or the power is unstable.

The ways that edge computing is measured have changed to match the needs of these systems that are spread out. With PureEdgeSim and other simulation tools, you can see how things work on a huge scale [18].

Researchers can use it to build models and test different settings for cloud, edge, and mist computing. These simulators can look at different architectures, orchestration approaches, and how offloading criteria affect things like latency, energy use, and how often tasks are finished. Researchers are now looking at how to quickly characterize the performance of massive, weakly connected, and geographically spread-out systems.

High-performance computing uses certain benchmarks, such High-Performance Conjugate-Gradient (HPCG), to show how scientific applications consume data and do math. This has an effect on how systems are made. In intelligent edge computing,

machine learning models need to be able to handle attacks from bad actors as well as be accurate and efficient. For instance, a Digital Twin design that uses federated learning leverages anomalous model recognition to improve both its total iteration delay and its loss function at the same time. This makes the DT fast, precise, and safe. This form of benchmarking shows you everything you need to know about how well the system works in tough edge cases.

How to keep your data and privacy safe

Edge computing needs to keep people's private information safe, especially as smart surroundings collect and look at it. Edge systems are made to look at data that is closer to where it came from. This makes it easier and harder to keep people's private information safe. There should be good techniques to keep people from getting to, distributing, or using data in ways that aren't allowed because there is so much of it from so many different areas. In the real world, edge nodes could not be extremely robust.

People not knowing where they are is a big problem. Fixed edge devices can do edge computing with IoT devices that are close by. Automatic logs of connections and authentication keep track of where IoT devices are, which means that location data could be compromised. Researchers are trying to find ways to make authentication difficult so that this problem can be solved. For instance, a system that uses a two-user ring signature for quick and easy authentication can stop location leaks by making it hard for anyone outside the system to prove communication transcripts. This saves between 10.728% and 14.696% of the costs of computing compared to alternative ways to refuse authentication that perform just as well.
When edge computing is used, privacy-enhancing techniques (PETs) are particularly vital for protecting data. Some examples are private information retrieval (PIR), secure multiparty computation, and ways to clean up data. But these solutions often need a lot of computing power, which edge devices with limited resources may not be able to provide. Two more basic approaches to keep your privacy safe are differential privacy and homomorphic encryption. They are used to protect private information while it is being collected, in point-of-interest services, and in apps that let users work together to get things done.

Federated Learning (FL) could be a technique to execute machine learning on the edge without giving up your privacy. FL trains models on every device and then sends changes to all of the models at simultaneously. This implies you don't have to keep all of the raw data in one place. This plan is great for IoT programs that need to keep information private, such smart homes or healthcare. But FL still needs good ways to encrypt and manage keys to keep model parameters safe while they are being sent and to stop data from leaking during training. For instance, a group key agreement mechanism can be used to encrypt and decrypt data that IoT devices send to each other. This adds hidden attribute authentication and enables you access more than one policy, which makes your privacy better.

In edge computing, it's still hard to achieve the right balance between keeping users' data private and making it helpful. Differential privacy and anonymization techniques try to strike a balance between protecting people's privacy and making data valuable. We need solutions that can work together to deal with the differences, scalability, and resource limits that are common in edge contexts [8][12] if we wish to move forward.

Talk and Look Over

The last study talks about how edge computing has changed over time and the issues with privacy, security, and performance in smart homes. This section delves into more detail on the complicated linkages and challenges, looking closely at the pros and cons and possible future courses.

To illustrate the practical applicability of the reviewed frameworks, a small case study on Energy Consumption Optimization in Edge Computing using PureEdgeSim:

A Java-based simulator constructed as an extension of CloudSim, to show how simulation tools documented in the literature are used to evaluate edge computing scenarios. This example provides context rather than an experimental evaluation.

Typical smart surveillance topologies contain six video camera–based IoT devices, two edge servers at local control centers, and a cloud data center. Assuming real-time object detection, video frames are generated and analyzed frequently.

The primary setup parameters were chosen based on representative values from previous studies. Video frames were generated every two seconds and required 1500 million instructions (MI). Edge servers were designed with 2000 MIPS, while the cloud data center had 10,000 MIPS. Network latency was anticipated to be 3 ms between cameras and edge servers and 90 ms between edge servers and the cloud.

For demonstration, two common processing strategies were considered:

a) Cloud-centric processing sends all video frames to the cloud data center for analysis.

b) Edge-assisted processing, where edge servers process video frames and provide only summary results or alarms to the cloud.

Observations of Representative Output

Energy, processing delay, and network use measurements were output by the simulator.

Metric	Cloud-only	Edge-enabled

Average Energy Usage (J)	320	105
Average Processing Latency (ms)	110	12
Network Usage (MB/min)	150	40

These outputs qualitatively show cloud-centric and edge-assisted execution model system behavior differences.

The typical outputs show that edge-assisted processing reduces data transmission and processing delays compared to cloud-centric processing. Multiple edge computing research reveal similar tendencies. So, the case study shows how PureEdgeSim is used to examine resource distribution and execution strategies in edge-enabled IoT systems.

Edge computing: how to strike the proper balance between speed and safety

Finding the right balance between security and speed is a big problem when using edge computing frameworks in smart environments. On the one hand, strong security measures are necessary to protect private information and keep the system running smoothly. On the other hand, these steps often add extra work to computers and communications, which can cause longer delays, more energy use, and less throughput, especially on edge devices that don't have a lot of resources.

Encryption is important for keeping data private, but it can make processing much slower and use up a lot of device resources. Homomorphic encryption and other advanced cryptographic algorithms offer robust protection, but they are not practical for many real-time edge applications because they need a lot of computing power. In the same way, secure communication protocols that need authentication use more bandwidth and have more latency than unsecured connections.

Access control, intrusion detection, and anomaly detection are other important security measures that can be quite accurate but need a lot of processing power. Machine learning (e.g., LGBM, OC-SVM) is typically used to power these measures. Putting these models into use in real time could slow down applications that need low latency. Designers need to think carefully about the pros and cons of attack avoidance and how it can slow down the system's response time.

Also, problems with integration make these problems worse. Smart environments are naturally diverse because they include devices from different manufacturers that use different hardware, operating systems, and communication protocols. This variety makes it harder to manage devices, enforce security regulations, and share data easily. Scalability is another important issue. Smart cities may need to handle millions of devices, whereas smaller areas may just need to support a few. Common ways to scale cloud services don't always work for edge deployments that are spread out and don't have a lot of resources. This means that dynamic, flexible resource management solutions are needed.

Interoperability is still a big problem. It is hard to move data and install applications smoothly on different devices, edge nodes, and cloud services since there are no standard protocols or APIs. This problem gets worse when data from several smart sectors (like energy and transportation) needs to be combined for a full analysis.

Open standards and abstraction layers are becoming more popular as a way to fix these problems with integration and interoperability. Software-Defined Networking (SDN), containerization (like Docker and Kubernetes), and standardized APIs are all technologies that make it easier to program and deploy across different types of hardware. AWS IoT Core and Greengrass are examples of commercial platforms that try to connect a wide range of IoT devices to cloud services that can grow and are safe.

In short, smart environments need edge computing frameworks that can handle the complicated relationship between security, performance, and integration. Solutions frequently include spreading security chores out based on what each device can do, using both hardware- and software-based protections, and changing security levels based on how important and risky an application is. In the end, this area will only be successful if standards are developed together, architecture is designed to be flexible, and trade-offs are always evaluated to make sure that smart systems are safe, fast, and can work with other systems.

Framework	Key Features	Security Techniques	Performance Metrics	References
EdgeX Foundry	Open-source, microservices, device mgmt	TLS, API authentication, RBAC, PKI	Latency, Throughput, Scalability, CPU	[EdgeX Foundry Docs]
AWS Greengrass	Cloud integration, ML at edge	Data encryption, IAM, secure boot	Response Time, Bandwidth Usage, Reliability	[AWS Greengrass Whitepaper]
Azure IoT Edge	Modular, supports Docker, cloud sync	Hardware-based security, TPM, secure comms	Latency, Resource Overhead, Scalability	[Microsoft Docs]
FogBus	Lightweight, blockchain integration	Blockchain, OAuth 2.0, secure MQTT	Delay, Energy Consumption, QoS	[FogBus Research Paper]
OpenFog	Interoperability, distributed control	Multi-layer security, access control	Latency, Jitter, Packet Loss	[OpenFog Consortium]

Cloudlet	VM-based, offload computation	Isolation, sandboxing, encrypted channels	Task Completion Time, Energy Savings	[Cloudlet Research]
KubeEdge	Kubernetes extension, edge-native	Node authentication, TLS, RBAC	Fault Tolerance, Latency, Scalability	[KubeEdge Docs]

Problems with integration: disparities, scalability, and working together

Smart environments that use edge computing frameworks are hard to set up and run since they have a lot of integration issues. Most of these challenges are caused by the fact that systems need to function together, systems need to be able to evolve, and technology is different. A smart environment is one where devices from different firms may function together, even if they have different hardware, operating systems, and ways of talking to each other. This built-in variety makes it tougher to keep track of devices, gather data, and make sure that security regulations are always followed throughout the ecosystem. Another important difficulty is how well it can expand. As smart environments get better, they can link to more and more devices. A smart city could have millions of connected gadgets, yet a home might only have a few sensors. Edge computing frameworks need to be able to change size and shape to handle new devices and workloads without slowing down or crashing. Standard scaling methods that work well for cloud data centers that are located in one place don't work right away on the edge, which is spread out and has limited resources. You need flexible resource orchestration and management solutions for this, as shown by simulation tools like PureEdgeSim, which can test multiple resource management strategies in changing environments.

Interoperability, or the ability for diverse systems and devices to function together without any difficulties, is just as hard. There are no standard means for edge devices, edge nodes, and cloud services to talk to each other, use data, or connect to each other. This makes it hard to easily migrate data and set up apps. It's considerably harder to achieve this with cross-domain apps, which need to incorporate data from different smart environment sectors, like smart energy and smart transport, for a thorough analysis. It's harder to figure out how the system works and costs more to maintain it running because there isn't a clear way to deal with all of these diverse elements.

Two typical strategies to fix these integration challenges are to use open standards and make abstraction layers. For example, people are looking into Software-Defined Networking (SDN) and other similar technologies as ways to make it easier to control multiple IoT devices and edge servers by giving them all the same programmable interfaces. This would make it simple to link PCs and share files. Containerization solutions like Docker and Kubernetes also make it easier to install software on a wide range of edge hardware in a way that works every time. This makes it easier to maneuver and control the apps. It might also be easier for all the services and platforms to work together if they all use the same Application Programming Interfaces (APIs) and data models. AWS IoT Core and Greengrass are two technologies that strive to connect different IoT devices to cloud services in a way that is both safe and can grow. To make edge computing perform optimally in dynamic smart environments, several integration problems need to be fixed.

Bibliography

1. W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, “Edge Computing: Vision and Challenges,” IEEE Internet of Things Journal, vol. 3, no. 5, pp. 637–646, Oct. 2016. doi: 10.1109/JIOT.2016.2579198
2. M. Satyanarayanan, “The Emergence of Edge Computing,”Computer, vol. 50, no. 1, pp. 30–39, Jan. 2017.doi: 10.1109/MC.2017.9
3. Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, “A Survey on Mobile Edge Computing: The Communication Perspective,” IEEE Communications Surveys & Tutorials, vol. 19, no. 4, pp. 2322–2358, 2017. doi: 10.1109/COMST.2017.2745201
4. P. Mach and Z. Becvar, “Mobile Edge Computing: A Survey on Architecture and Computation Offloading,”IEEE Communications Surveys & Tutorials, vol. 19, no. 3, pp. 1628–1656, 2017.doi: 10.1109/COMST.2017.2682318
5. T. Taleb et al., “On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Cloud Architecture,” IEEE Communications Surveys & Tutorials, vol. 19, no. 3, pp. 1657–1681, 2017. doi: 10.1109/COMST.2017.2705720

6. F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, “Fog Computing and Its Role in the Internet of Things,” in Proc. MCC Workshop, 2012, pp. 13–16. doi: 10.1145/2342509.2342513
7. X. Sun and N. Ansari, “EdgeIoT: Mobile Edge Computing for the Internet of Things,” IEEE Communications Magazine, vol. 54, no. 12, pp. 22–29, Dec. 2016. doi: 10.1109/MCOM.2016.160037Edge
8. R. Roman, J. Lopez, and M. Mambo, “Mobile Edge Computing, Fog et al.: A Survey and Analysis of Security Threats and Challenges,” Future Generation Computer Systems, vol. 78, pp. 680–698, Jan. 2018. doi: 10.1016/j.future.2016.11.009
9. L. Xiao, Y. Li, G. Han, G. Liu, and W. Zhuang, “PHY-Layer Spoofing Detection With Reinforcement Learning in Wireless Networks,” IEEE Transactions on Vehicular Technology, vol. 65, no. 12, pp. 10037–10047, 2016. doi: 10.1109/TVT.2016.2519504
10. Z. Guan et al., “Privacy-Preserving Authentication for Edge Computing,” IEEE Transactions on Dependable and Secure Computing, vol. 17, no. 3, pp. 515–528, May–June 2020. doi: 10.1109/TDSC.2018.2872807
11. K. Zhang et al., “Blockchain-Based Secure Data Sharing System for Internet of Vehicles,” IEEE Transactions on Vehicular Technology, vol. 68, no. 4, pp. 4187–4201, Apr. 2019. doi: 10.1109/TVT.2019.2893122
12. M. Conti, A. Dehghantanha, K. Franke, and S. Watson, “Internet of Things Security and Forensics: Challenges and Opportunities,” Future Generation Computer Systems, vol. 78, pp. 544–546, 2018. doi: 10.1016/j.future.2017.07.060
13. J. Konečný et al., “Federated Learning: Strategies for Improving Communication Efficiency,”arXiv:1610.05492, 2016.
14. Q. Yang, Y. Liu, T. Chen, and Y. Tong, “Federated Machine Learning: Concept and Applications,” ACM Transactions on Intelligent Systems and Technology, vol. 10, no. 2, 2019. doi: 10.1145/3298981
15. S. Wang et al., “Dynamic Service Placement for Mobile Micro-Clouds with Predicted Future Costs,” IEEE Transactions on Parallel and Distributed Systems, vol. 28, no. 4, pp. 1002–1016, 2017. doi: 10.1109/TPDS.2016.2603520
16. Y. Mao, J. Zhang, and K. B. Letaief, “Dynamic Computation Offloading for Mobile-Edge Computing With Energy Harvesting Devices,” IEEE Journal on Selected Areas in Communications, vol. 34, no. 12, pp. 3590–3605, 2016. doi: 10.1109/JSAC.2016.2611964
17. A. Yousefpour et al., “All One Needs to Know About Fog Computing and Related Edge Computing Paradigms,” Journal of Systems Architecture, vol. 98, pp. 289–330, 2019. doi: 10.1016/j.sysarc.2019.02.009
18. M. T. Beck et al., “Mobile Edge Computing: A Taxonomy,” IETF Draft, 2014.
19. H. Gupta et al., “iFogSim: A Toolkit for Modeling and Simulation of Resource Management Techniques in IoT,” Software: Practice and Experience, vol. 47, no. 9, pp. 1275–1296, 2017. doi: 10.1002/spe.2509
20. S. Yi, Z. Hao, Z. Qin, and Q. Li, “Fog Computing: Platform and Applications,” in Proc. IEEE HotWeb, 2015, pp. 73–78. doi: 10.1109/HotWeb.2015.22

References for Table

a. Shi, W. et al. (2016). Edge Computing: Vision and Challenges. IEEE IoT Journal.
b. Varghese, B., & Buyya, R. (2018). Next Generation Cloud Computing: New Trends and Research Directions. FGCS.
c. EdgeX Foundry Documentation: https://www.edgexfoundry.org/
d. AWS Greengrass Documentation: https://docs.aws.amazon.com/greengrass/
e. Azure IoT Edge Documentation: https://docs.microsoft.com/en-us/azure/iot-edge/
f. KubeEdge Documentation: https://kubeedge.io/