Reliable Edge Computing Architectures for Crowdsensing Applications

Timothy Thomas George
School of Computer Science and Engineering,
Vellore Institute of Technology, Chennai Campus,
Chennai, 600127, Tamilnadu, India.
timothythomas.george2020@vitstudent.ac.in

Amit Kumar Tyagi[0000-0003-2657-8700]
School of Computer Science and Engineering,
Vellore Institute of Technology, Chennai Campus,
Chennai, 600127, Tamilnadu, India.
amitkrtyagi025@gmail.com

Abstract. Edge computing has opened the door to a wide range of opportunities in the field of crowdsensing. Processing that happens close to the end devices has many advantages over those that take place at the cloud, such as lower latency and near real-time computation. However, research on the reliability of edge-computing paradigms is important if we wish to maintain the Quality of Service (QoS). This paper aims to review research studies that emphasize reliability in edge-computing architectures. We will discuss various approaches that researchers have taken to tackle resilience in Internet of Things (IoT) networks. Thus, we will be able to gauge the status of IoT reliability research. We have divided the research studies based on the approach taken by researchers in proposing suitable models. The majority of literature analyzes networks based on layers, such as edge, fog, cloud, etc. These proposed architectures can support edge-computing practices. Crowdsensing can utilize reliable edge-computing architectures to develop improved systems and procedures. Studies on reliability are crucial since edge-computing architectures have found applications in mission-critical areas such as healthcare. Reliability studies is also important for the manufacture of energy-efficient systems. This is because if a system has a reliable design, then less time will be spent on its maintenance. Hence, the applications of this study are not limited to crowdsensing.

Keywords: Edge Computing, Reliability, Internet of Things Architecture, Fault-Tolerance, Crowdsensing Applications

I. INTRODUCTION

Typical IoT architecture requires a network of mainly three components – sensors, devices and servers [1]. Sensors are responsible for collecting data from the environment. The data is then sent to connected devices, which in turn sends it to the server where it is processed. Crowdsensing implements similar networks, on a larger scale. It depends on data sensed from 'the crowd' which often includes an entire neighborhood or a city. However, the three segments are typically far away from each other, with servers being the furthest. The rise of edge computing as an alternative to cloud computing aimed to bring these three components closer, which was achieved by doing computation near the sensors, or at the 'edge'. Today, crowdsensing technology applies the existing edge-computing knowledge. Therefore, many researchers have been working to improve the edge-computing methods.

Edge computing has many benefits, the most significant of these is the lower latency of communication. Cloud computing necessitated the transfer of data from the end-devices to the cloud, where analysis and computation could be done. With edge computing, the processing is done faster, and only the computation that the edge network is incapable of completing is sent to the cloud servers. There are many applications where we simply do not have the time for cloud computation, and where we rather need results in real-time. An example of this is space exploration. The sensors that detect abnormalities in a spacecraft's trajectory must immediately rectify them by an appropriate response, or else the effect could be catastrophic. Such applications are called 'mission-critical' applications. When IoT applications are affected by the delay caused to transfer data, edge computing offers a good workaround.

Manufacturers, regardless of their product, must ensure that their product is reliable. This is because reliability is a measure of the quality of a product [2]. Similarly, if any system that is concerned with having a high Quality of Service (QoS) must include the reliability of its system as a topmost priority [2]. For a particular application to meet an expected QoS requirement, it is crucial to make the right choice of architecture. Edge computing architecture poses unique challenges to manufacturers. They must be kept reliant even if they possess lesser computational capability than the cloud infrastructure. In case if there is a security failure at the cloud level, the exploitation of the edge layer must be prevented [3]. Thus, it is important to analyze and research reliable edge-computing architectures.

The importance of reliable IoT networks has grown since IoT has found application in such mission-critical processes [4]. These scenarios require both low latency and high availability. IoT devices themselves have been developing quickly, propelled by the prediction of Moore's Law [4]. It has allowed for the manufacturing of smaller and more efficient components of IoT networks. IoT devices can today be used in a wide variety of areas, such as smart cities and homes, but can also constitute domain-independent services, such as data analytics [5]. But edge-computing is not without problems. The major issues of concern were related to privacy, and changing contexts. This is because most of today's crowdsensing application is based on user-owned devices. This serves as an alternative to explicit large-scale IoT device deployment.

The increase in popularity of mobile crowdsensing can also be partly due to the sudden increase in the number of mobile devices in the environment. In the past, mobile crowdsensing had to be implemented by distributing mobile devices to participants who took part in the experiment. Today, research on mobile crowdsensing can be done using personally owned mobile devices. This greatly reduces the cost, since the manufacturing of end IoT devices is no longer required. Hence, they can be replacements for the sensors in Wireless Sensor Networks (WSN) [6]. There are two types of problems associated with mobile computing – system faults or user faults. System faults can be corrected by a certain reliable architecture being chosen. The majority of the paper will discuss these architectures, finally we through some light on user faults.

This paper attempts to discuss reliable edge-computing architectures and is organized in the following manner. Section II reviews the current literature behind the concepts of reliability and fault-tolerance. Layer-based architectures are discussed in Section III. In Section IV we will analyze the other network architectures that aim to be reliable by grouping them into broad subcategories. In Section V, we will discuss the current research in the domain of reliable mobile crowdsensing. In Section VI, we will describe the limitations of this review, as well as outline the scope of future research. Finally, we will conclude this paper with a few implications of this study.

II. RELATED WORK

A. Reliability

Studies in Reliability first found their use in failure prediction, when it became important to determine when a piece of equipment would cease operating [7]. The different notions of reliability are discussed in [8]. That reliability guarantees robustness when faced with an error is called the 'classical notion.' This classical notion implies that high reliability will be compensated by increased delays present in the network. The methods proposed in [8] to ensure high reliability were multi-connectivity, task replication, federated machine learning, and extreme event control. The other notion is concerning a correct result being produced within a time-bound, which is as discussed in [9].

Reliability in [1] was brought about by the replication of data on the edge of the network. [11] has classified two types of replication strategies, viz. active replication, and passive replication. Active replication involves the continuous execution of a process concurrently by different parts of the network. However, in passive replication, a process is executed only by the main component, while the other components wait for a failure to occur before being utilized. This method is slower but takes less processing resources during execution. Today, reliability is considered to be an important requirement in almost every field and is closely linked to fault-tolerance, which is discussed next.

B. Fault Tolerance

According to [12], fault tolerance is one of the three major strategies for dealing with faults. The other two are fault correction and fault avoidance. In a study of the current research of fault-tolerant IoT, [11] has arrived at many important conclusions. Much of the research conducted in the field of cloud security extends into the fog and cloud realm. Secondly, the measure of distribution, collaboration, and intelligence of and between components in a network had a great impact on the resilience of the system. According to [10] a partially operating fault tolerance mechanism can prevent a system from shutting after a failure, and instead cause it to run at a lower capacity than before.

Fault tolerance can be broadly classified into two main categories – reactive fault tolerance and proactive fault tolerance [9]. Reactive fault tolerance takes place when a failure occurs and the system 'reacts' to minimize the damage caused. There are many methods of reactive fault tolerance, such as checkpointing, replication, etc. In [1] fault tolerance was brought about by redirection. When a failure was encountered, the server is redirected in the most efficient way possible in the given network hierarchy. This process needs to be done automatically, without the requirement of user instruction. [9] Proactive Fault Tolerance implies that the system predicts the faults and acts 'proactively' to replace components deemed as suspicious. Proactive fault tolerance methods include preemptive migration and self-healing. As suggested in [1], one way of achieving fault tolerance is to deploy architectures that are capable of self-management, giving itself an 'intelligence'.

We must bring reliability and fault tolerance to the fog layer of the services. [13]. This is because the fog layer is vulnerable, since it may not enjoy strong protection available to the cloud layer. Thus, failures can be caused by an unusual load, whether that extra burden is brought about legally (Big Data) or illegally (DDoS Attacks) [13]. There are many vulnerabilities in IoT [11], based on different components of the network being missed or malfunctioning. IoT devices are usually powered by a battery and are wireless, and are constantly witnessing new services. All the features increase the complexity of designing a fault-tolerant and reliable IoT architecture. Nevertheless, many researchers have proposed different architectures for reliable IoT architecture, which are discussed in the subsequent section.

III. LAYER-BASED ARCHITECTURAL APPROACH

In a survey of the different IoT architectural patterns for fault-tolerance in [11], it was found that the favorite among researchers was the layered architecture. This was followed by the cloud-based architecture. In [12], the authors discuss that a multi-level configuration is essential in a robust application. Some of the researchers describe what is called a 'fog' layer. It was in 2012 that Cisco introduced the concept of fog computing. [4]. Though often confused with, fog and edge computing are distinct, as discussed in [13]. Fog computing implies that processing is brought down to the LAN level of the network, therefore intelligence is close to the ground, like a fog. It refers to a level in between the edge (devices) and the cloud (server). Edge computing gives smart devices themselves the computational power required to process data. Edge computing sought to utilize the infrastructure present at the edge of the network for computational and storage processes [14].

The architecture in [1] proposed a four-layer approach, i.e., Cloud-fog-mist-dew. The dew layer is comprised of the actual IoT devices. Any computation that happens at this layer took place to produce outputs that were required by the application in real-time. The mist layer, also known as the roof layer, consists of a server close to the IoT layer. The fog layer bridges the gap between the cloud and mist layers. The authors in [1] say that in the real world, these servers are likely to be deployed by an ISP, but can also exist separately in a network. The computation ability is greatest in the cloud layer, which may hold the actual data and operational centers. [4] also acknowledges how correct results can be obtained in reduced time if we use multiple layers in each part of the architecture. Each layer can give intermediate results in the process of error detection, with different stages completed at subsequent layers.

A four-level approach is proposed in [15]. These levels are – Embedded layer, multi-server layer, the core layer, and cloud layer. The embedded layer consists of sensors that are deployed in the environment for a particular function. The second later is where services are available, similar to the 'fog' layer that other researchers have talked about. The core layer is where various technologies and requirements are checked to verify Quality of Service (QoS). The resource-heavy computations, such as storage of data across a long timeline, are done at the cloud layer. This architecture used containers for integration and was implemented using a Mininet emulation tool. However, as discussed in [12], interdependency among containers may cause some issues in the application, and this also needs to be taken care of.

Some researchers have surveyed architectures of a particular layer configuration, which others have focused their study on one particular layer. [13], in their survey, surveyed a three-layer approach- edge, fog, and cloud. A fault-tolerant session layer has been proposed in [16]. Though TCP is the most popular communication protocol, it does have some limitations. In case of a failure, the endpoints may not be able to determine whether a message has reached a destination, due to which the reliability is insufficient to coordinate peer-dependent processes. The proposed layer in [16] can track messages being sent, allowing it to roll back in the event of a failure. A consensus protocol proposed in [14] is further discussed in subsequent sections.

The fault-tolerant architecture in [14] proposed 3 layers, viz. devices layer, edge layer, and a cloud layer. The device layer consists of mobile devices which handle application requests. The nodes present in the edge layer are responsible for the computation of certain processes. Finally, cloud services are provided at the cloud layer. Like in this case, the use of mobile phones in crowdsensing architecture has opened new avenues of research. Many researchers have begun working on architectures for crowdsensing that focus on enforcing the privacy of the users. However, these can only be implemented if there exists a reliable architecture on which to do so. This architecture must be able to operate even in hostile environments.

IV. OTHER ARCHITECTURAL APPROACHES

A. Special Components

The architecture as proposed by [4] prevents IoT applications from accessing devices directly. Instead, the IoT applications access virtual devices, called 'shadow devices. The very use of shadow devices implies that recovery is needed in case of any interference with the physical device. When recovery is required, the shadow devices are reconfigured automatically from another device. Therefore, the physical devices need not be changed or communicated to. The software used in [4] to implement this architecture consists of registers and connectors to record and monitor the devices respectively. Apache Kafka was utilized for quick message dispatches. [4] says that this particular architecture is 'application-agnostic with respect to IoT protocols. This type of recovery block where an alternative shadow node becomes active in case of a failure of the main node is also discussed in [11].

In [12], three components are present in the system, viz. self-configuration, which is responsible for accurate infrastructure based on the requirements, self-optimization, which is responsible for monitoring, and self-healing, which recovers faulty infrastructure. The three components may be present individually or distributed among many layers of architecture. Researchers in [12] combine both layered architecture and time-dependency to propose a model capable of recovery. To meet the latency requirements, cloud-based resources are utilized. When a failure occurs, the self-healing component switches off the affected resources and recovers them by switching on backup resources using recovery procedures. This is considered an energy-efficient methodology.

B. Protocols

A neural network-based procedure for failure detection has been outlined in [9], called the 'heartbeat strategy'. This strategy is algorithmic in approach. A process is recognized as 'suspicious' if a predetermined message or 'heartbeat' is not received from it after a fixed interval of time. The process can be stripped of its 'suspicious' status if it can send the message later, i.e., after the 'freshness points'. A similar type of architecture is discussed in [11]. The network is assigned a cluster head, which is responsible for contacting other nodes periodically. If a particular component does not respond, the head concludes that a failure has occurred at that portion of the system. This time-based approach is also used in the protocol solution to the consensus problem in [14], where if a message from a node is not received within the predicted time period, the influence of faulty components is confirmed.

The communication-centered protocol in [14] is based on bringing all the nodes present in the network to reach a consensus as to how many faulty components can be tolerated in the IoT network. The paper [14] introduces this issue of getting nodes to agree on a single value as the 'consensus problem'. It is a basic problem related to reliable transmission in distributed systems. [17]. The solution to the problem brought forth protocols. Protocols aim to make the nodes agree on the largest possible number of faulty components to tolerate while using the least amount of communication. One such protocol is proposed in the same paper.

For the protocol proposed in [14], the authors discuss the possibility of a fault in the medium of transmission and consider this as a node fault. Two types of faulty transmission mediums are classified – dormant, which can be predicted based on certain coding schemes, and malicious, which cannot be predicted. Since the system needs to be available even if the transmission mediums have been interfered with by an adversary, the nodes will use the protocol to come to the required consensus. Another method discussed in [11] is based on taking additional time to enforce redundancy to provide fault tolerance. There are two times of time redundancy, reactive, which reacts after error-detection, and proactive, which does not wait for error detection but predicts it through machine learning.

C. Microservices

Another analysis in [11] was that an increasing number of research papers were proposing microservices as solutions to improve the fault tolerance of IoT networks. Like its name, microservices are small components with small functions. The advantage of using microservices lies in the ease with which we can deploy and tests their reliability. [11] puts microservices-based architectures as Service-Oriented Architectures, where the services offered become central to the structure of the network. Asynchronous behavior of the system, that results in lower coupling, is essential in guaranteeing the resilience of the architecture [18]. Therefore, microservices are a good option for the components of a network. Reactive microservices [15] are known for their resilient and elastic attributes. These independent units can build up architectures effectively. The idea of using a micro-component instead of the entire component can be applied to other parts of the network. For example, in the proposed model given in [19], the payment process was made secure using micropayment services, which were implemented in the blockchain.

V. Mobile Crowdsensing

Mobile crowdsensing is another option to be considered. It depends on the sensors present on people's devices as the end IoT devices. Therefore, there may be less hardware deployment required. Hence, the dependence of the system on the hardware reliability is reduced, however, mobile crowdsensing also brings up other reliability-related issues. The users who are involved in mobile crowdsensing may be the source of faults. The processes that are related to mobile crowdsensing also need to be reliable. For example, in [19], to make the payment process that followed after the IoT device's main function was implemented using blockchain, which is made it more secure.

In a mobile-crowdsensing environment, a user may not be successful his executing his task on his mobile device [20]. There is a probability related to the chance of completing the task on the user's side. There are many reasons for failure on the user's side [20]. These are related to the inconsistent mobility patterns of the user or the fault of an unreliable network connection between the device and the remaining components of the system. There is even a chance that the user may provide incorrect information if they perceive some advantage in doing so. The devices themselves also have limits to the amount of computation that they will be capable of, due to their physical architecture.

Yet, with further analysis, we can correlate this to the fault-tolerance that we have discussed earlier. Users also represent components in the network that are unreliable. A problem that arises here is that this probability of success is information not accessible to the system. It again raises the issue of privacy in mobile crowdsensing, about which much research is going on. Therefore, researchers have come up with different methods to obtain this probability value. One of them [20] has discussed the use of machine learning algorithms, however, this is based on the assumption that what data is received from the user is true.

In the mechanism designed in [6], the design is of agents that are responsible for ensuring the probability of success (PoS) of activity at a desired high level, over a particular period of time. Then the focus shift to reduce the cost required for the agents is only high enough to guarantee that the required probability is met. Analogous to the hardware (also software) redundancy methods to ensure fault-tolerance, one way in guaranteeing fault tolerance in a crowdsensing network is by having enough users. To achieve this, more users need to be motivated toward the platform [20]. This opens the study of incentivization, which needs to be advantageous to both client and corporation.

VI. ISSUES, CHALLENGES AND OPPORTUNITIES FOR FUTURE RESEARCHERS

Although this paper attempted to provide an exhaustive overview of reliable edge-computing architectures, it is not without issues. This paper consisted of purely a theoretical review and was unable to delve into the performance of the edge-computing architectures at the practical level. We have, however, formed a conceptual foundation that can help us expect reliable performance from certain architecture. This paper has fallen short of putting these various architectures to the test in real-life situations. Environmental factors may affect the practical setting up of a particular architecture. As has been mentioned earlier, a reliable design must be able to tolerate hostile environments. Installing a reliable IoT architecture in an unfavorable environment would drive costs upward, and the financial aspect is another area this paper was not able to cover. Moreover, certain theoretical terminologies may be understood differently in the industrial world. As we have mentioned earlier, the terms 'fog-computing' and 'edge-computing' are sometimes confused with, and it should not come as a surprise that sometimes it is observed that these terms are used interchangeably. Nevertheless, for all research purposes, we will stick to the separate definitions as given in [13].

There were also some challenges faced during the course of this review. Firstly, we know that it is easier to propose new architectural models than to manufacture and distribute them. The implementation of a newly proposed architecture is a difficult task, especially if very little research has been conducted on it. Hence, the rate at which new models are proposed in the realm of research will be faster than the rate at which they are utilized in the industrial world. Thus, this causes a disparity to grow between the two groups, with industry behind the research. This cautions us against conducting wide-scale research on topics that gather interest only among researchers but have little to no significance outside academia. Furthermore, architectures alone do not guarantee reliable edge computing. Several other components, such as communication protocols, need to be appropriately selected. A good communication protocol can ensure reliable and constant performance, even if the physical devices employed in the architecture face mechanical errors. Due to these several non-architectural edge computing network constituents, this paper avoided selecting one 'best edge computing architecture' for use in mobile crowdsensing. Instead, different architectural approaches were described.

There is plenty of scope for more work in the domain of reliable edge computing architectures. Future research will need to tackle the challenge of developing future-proof standards. One approach to this is to ensure that the designs are able to evolve based on the new needs and requirements. We can expect, with reasonable certainty, that the demand for faster, yet more reliable computation will only increase in the years to come. Hence, long-term sustainability is one area that researchers need to look into. The near future will also see an increase in the number of personal devices connected to each other. These new devices pose new opportunities and challenges for both industry and academia alike. Moreover, our increased reliance on devices establishes the importance of reliable edge-computing architectures. Now, Table 1 and table 2 shows some impact of AI on Cloud services in detail.

Table 1. Impacts of AI on Services

Name	**Positive Impacts**	**Negative Impacts**
Impact on Healthcare	Creation of new drugs Improved Diagnostics	Human care being replaced by AI systems
Impact on Customer services	Customization of services	Replacement of humans
Impact on Transportation and Logistics	Non-stop autonomous transportation. Reduce car crashes Learning capabilities of autonomous vehicles	Autonomous cars will reduce the driving pleasure.
Impact on Education	New courses in school	-
Impact on Environment-related services	New service opportunities for societal well-being.	High electricity consumption from AI technologies
Impact on Financial Services	Fraud detection Disappearance of passwords for facial or digital print recognition	Lack of credibility of cryptocurrencies
Impact on Retail	Physical stores will still exist in the future for customer care.	Replacement of jobs

Finally, some crosscutting topics that impact all of the above services were also mentioned. These are shown in Table 2.

Table 2. Crosscutting Impacts of AI on Services

Name	**Positive Impacts**	**Negative Impacts**
Impact on Privacy	-	Loss of privacy Isolation
Impact on Labor	Smart redirection of Human Resources on society Repetitive tasks automation	Unemployment Replacement of jobs
Impact on Legal Aspects	Reduce the number of ID falsifications	No regulation of technologies Political issues Failure of explanation

Now, readers are recommended to know/ learn about emerging technologies like blockchain technology, edge computing, automated analytics, quantum machine learning, etc., and their uses in smart era, can be found in [21-30] in detail.

VII. CONCLUSION

A. Summary

The goal of this paper was to review edge computing architectures that emphasize reliability in supporting the latest edge-computing practices. These architectures must be able to operate correctly in various situations and different environments. Edge-computing, which evolved from the cloud-computing technology preceding it, has found applications in many areas. Crowdsensing is one such area that benefits from the advantages of the edge-computing paradigm. The reliability of a system is an important requirement for high quality; hence, research continues to progress in this domain. This paper aims to review those research studies that proposed or analyzed solutions that would make IoT-enabled edge-computing networks more resilient. As a result, we were able to get a bird's eye view of the current research direction in this area. Based on the approach taken by researchers we have divided them based on whether or not they proposed a layer-based model of reliability.

Other perspectives included those that sought reliability by including certain components, or a protocol of communication, or microservices. The majority of research has analyzed the networks based on layers, such as edge, fog, cloud, etc. Research is also taking place on another front, i.e., privacy preservation in crowdsensing. But, for this purpose, there needs to be a reliable architecture to work on. It is with this need in mind that the study was undertaken.

B. Outlook

In the proposed model in [12], The backup resources are switched off until they are needed, saving energy. We have discussed earlier that making a system more reliable involves more energy in terms of its development process. However, a more reliable system design can save manufacturing energy during maintenance. This is especially true in the case of self-healing components discussed earlier. More energy implies a greater environmental impact; hence we must continue research in this domain to help us reduce our environmental footprint.[13]. The authors in [15] agree that more research is required in the specific field of IoT reliability, such as in dynamic resource discovery.

With the increased application of edge computing in various disciplines, it is crucial that we study and further the research of reliable architectures. These architectures can then be utilized in crowdsensing and beyond.

REFERENCES

1. Grover, J., Garimella,R. M.: Reliable and Fault-Tolerant IoT-Edge Architecture. In: 2018 IEEE SENSORS, pp. 1-4. IEEE, New Delhi (2018). doi: 10.1109/ICSENS.2018.8589624.
2. White, Gary, Nallur, Vivek, Clarke, SiobhÃạn: Quality of service approaches in IoT: A systematic mapping. Journal of Systems and Software 132, 186 – 203 (2017). doi: 10.1016/j.jss.2017.05.125.
3. Han, B., Wong, S., Mannweiler, C., Crippa, M., Schotten, H.: Context-Awareness Enhances 5G Multi-Access Edge Computing Reliability. In: IEEE Access, vol. 7, pp. 21290-21299. IEEE (2019). doi: 10.1109/ACCESS.2019.2898316.
4. Martin, C., Garrido, D., Diaz, M., Rubio, B.: From the Edge to the Cloud: Enabling Reliable IoT Applications. In: 2019 7th International Conference on Future Internet of Things and Cloud (FiCloud), pp. 17-22. IEEE, Istanbul (2019). doi: 10.1109/FiCloud.2019.00011.
5. Shahid, N. and Aneja, S.: Internet of Things: Vision, application areas and research challenges. In: International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), pp. 583–587. IEEE, Palladam, India. doi: 10.1109/I-SMAC.2017.8058246.
6. Liu, Y., Zheng, Z., Wu, F., Gao, X., Chen, G.: Fault tolerant mechanism design for time coverage in crowdsensing system. In: 2017 IEEE 36th International Performance Computing and Communications Conference (IPCCC), pp. 1-8. IEEE, San Diego (2017). doi: 10.1109/PCCC.2017.8280460.
7. Singh, C., Billinton, R.: System reliability, modelling and evaluation, vol 769. Hutchinson London (1977).
8. Elbamby, M. S., Perfecto, C., Liu, Chen-Feng, Park, J., Samarakoon, S., Chen, X., Bennis, M.: Wireless Edge Computing with Latency and Reliability Guarantees. In: Proceedings of the IEEE, 107(8), pp. 1717-1737. IEEE (2019). doi: 0.1109/JPROC.2019.2917084.
9. Amin, Z., Singh, H., Sethi, N.: Review on Fault Tolerance Techniques in Cloud Computing", International Journal of Computer Applications, 116(18), 11-17 (2015).
10. Kaushal, V., Bala, A.: Autonomic fault tolerance using haproxy in cloud environment. International Journal of Advanced Engineering Sciences and Technologies, 7(2), 54-59 (2011).
11. Moghaddam M.T., Muccini H.: Fault-Tolerant IoT. In: Calinescu R., Di Giandomenico F. (eds) Software Engineering for Resilient Systems. SERENE 2019. Lecture Notes in Computer Science, vol 11732. Springer, Cham (2019). doi: 10.1007/978-3-030-30856-8_5.
12. Almurshed, O., Rana, O., Li, Y., Ranjan, R., Jha, D. N., Patel, P., Jayaraman, P. P., Dustdar, S.: A Fault Tolerant Workflow Composition and Deployment Automation IoT Framework in a Multi Cloud Edge Environment. IEEE Internet Computing, 1-1. IEEE (2021). doi: 10.1109/MIC.2021.3078863.
13. Maciel, P., Dantas, J., Melo, C., Pereira, P., Oliveira, F., Araujo, J., Matos, R.: A survey on reliability and availability modeling of edge, fog, and cloud computing. J Reliable Intell Environ. (2021). doi: 10.1007/s40860-021-00154-1.
14. Wang, S., Hsiung, W., Hsieh, C.:Reliability Enhancement of Mobile Edge Computing for the IoT with Multiple Damage Communication. In: Liu Y., Wang L., Zhao L., Yu Z. (eds) Advances in Natural Computation, Fuzzy Systems and Knowledge Discovery. ICNC-FSKD 2019. Advances in Intelligent Systems and Computing, vol 1075. Springer, Cham (2020).
15. Santana, C., Andrade, L., Mello, B., Batista, E., Sampaio, J., Prazeres, C.: A reliable architecture based on reactive microservices for IoT applications. In: Proceedings of the 25th Brazillian Symposium on Multimedia and the Web, pp. 15-19. (2019). doi: 10.1145/3323503.3345027.
16. Ivaki, N., Boychenko, S., Araujo, F.: A Fault-Tolerant Session Layer with Reliable One-Way Messaging and Server Migration Facility. In: 2014 IEEE 3rd Symposium on Network Cloud Computing and Applications (ncca 2014), pp. 75-82. IEEE, Rome (2014). doi: 10.1109/NCCA.2014.20.
17. Fischer, M. J.: The consensus problem in unreliable distributed systems (a brief survey). In: Karpinski M. (eds) Foundations of Computation Theory. FCT 1983. Lecture Notes in Computer Science, vol 158. Springer, Berlin, Heidelberg (1983). doi: 10.1007/3-540-12689-9_99.
18. Bonér, J.: The Evolution of Microservices at Scale. Reactive Microsystems. O'Reilly Media, Gravenstein Highway North, Sebastopol (2017).
19. Shi, F., Qin, Z., Wu, D., McCann, J.: MPCSToken: Smart Contract Enabled Fault-Tolerant Incentivisation for Mobile P2P Crowd Services. In: 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS), pp. 961-971. IEEE, Vienna (2018). doi: 10.1109/ICDCS.2018.00097.
20. Zheng, Z., Yang, Z., Wu, F., Chen, G.: Mechanism Design for Mobile Crowdsensing with Execution Uncertainty. In: 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), pp. 955-965. IEEE, Atlanta (2017). doi: 10.1109/ICDCS.2017.230.
21. M. M. Nair, A. K. Tyagi and N. Sreenath, "The Future with Industry 4.0 at the Core of Society 5.0: Open Issues, Future Opportunities and Challenges," 2021 International Conference on Computer Communication and Informatics (ICCCI), 2021, pp. 1-7, doi: 10.1109/ICCCI50826.2021.9402498.
22. Tyagi A.K., Fernandez T.F., Mishra S., Kumari S. (2021) Intelligent Automation Systems at the Core of Industry 4.0. In: Abraham A., Piuri V., Gandhi N., Siarry P., Kaklauskas A., Madureira A. (eds) Intelligent Systems Design and Applications. ISDA 2020. Advances in Intelligent Systems and Computing, vol 1351. Springer, Cham. https://doi.org/10.1007/978-3-030-71187-0_1
23. Goyal, Deepti & Tyagi, Amit. (2020). A Look at Top 35 Problems in the Computer Science Field for the Next Decade. 10.1201/9781003052098-40.
24. Varsha R., Nair S.M., Tyagi A.K., Aswathy S.U., RadhaKrishnan R. (2021) The Future with Advanced Analytics: A Sequential Analysis of the Disruptive Technology's Scope. In: Abraham A., Hanne T., Castillo O., Gandhi N., Nogueira Rios T., Hong TP. (eds) Hybrid Intelligent Systems. HIS 2020. Advances in Intelligent Systems and Computing, vol 1375. Springer, Cham. https://doi.org/10.1007/978-3-030-73050-5_56
25. Tyagi, Amit Kumar; Nair, Meghna Manoj; Niladhuri, Sreenath; Abraham, Ajith, "Security, Privacy Research issues in Various Computing Platforms: A Survey and the Road Ahead", Journal of Information Assurance & Security. 2020, Vol. 15 Issue 1, p1-16. 16p.
26. Amit Kumar Tyagi, S U Aswathy, Autonomous Intelligent Vehicles (AIV): Research statements, open issues, challenges and road for future, International Journal of Intelligent Networks, Volume 2, 2021, Pages 83-102, ISSN 2666-6030. https://doi.org/10.1016/j.ijin.2021.07.002.
27. Shabnam Kumari, Amit Kumar Tyagi, Aswathy S U, "The Future of Edge Computing with Blockchain Technology: Possibility of Threats, Opportunities and Challenges", in the Book "Recent Trends in Blockchain for Information Systems Security and Privacy", CRC Press, 2021.
28. A. K. Tyagi and N. Sreenath, "Providing trust enabled services in vehicular cloud computing," 2016 International Conference on Research Advances in Integrated Navigation Systems (RAINS), 2016, pp. 1-7, doi: 10.1109/RAINS.2016.7764391.
29. Rekha G., Tyagi A.K., Anuradha N. (2020) Integration of Fog Computing and Internet of Things: An Useful Overview. In: Singh P., Kar A., Singh Y., Kolekar M., Tanwar S. (eds) Proceedings of ICRIC 2019. Lecture Notes in Electrical Engineering, vol 597. Springer, Cham. https://doi.org/10.1007/978-3-030-29407-6_8
30. Amit Kumar Tyagi, G. Aghila, "A Wide Scale Survey on Botnet", International Journal of Computer Applications (ISSN: 0975-8887), Volume 34, No.9, pp. 9-22, November 2011.