Contents lists available at Science-Gate

International Journal of Advanced and Applied Sciences

Journal homepage: http://www.science-gate.com/IJAAS.html

Dependability in fog computing: Challenges and solutions

Sara Alraddady [1,*], Alice Li [1], Ben Soh [1], Mohammed AlZain [2]

[1]La Trobe University, Melbourne, Australia
[2]College of Computers and Information Technology, Taif University, Taif, Saudi Arabia

ARTICLE INFO

Article history:
Received 4 October 2020
Received in revised form
20 December 2020
Accepted 23 December 2020

Keywords:
Fog computing
Fault tolerance
Availability
Placement policy

ABSTRACT

The tremendous increase in IoT devices and the amount of data they produced is very expensive to be processed at cloud data centers. Therefore, fog computing was introduced in 2012 by Cisco as a decentralized computing environment that is considered to be more efficient in handling such a plethora in the number of requests. Fog computing is a distributed computing paradigm that focuses on bringing data processing at the network peripheral to reduce response time and increase the quality of service. Dependability challenges of such distributed and heterogeneous computing environments are considered in this paper. Because fog computing is a new computing paradigm, several studies have been presented to tackle its challenges and issues. However, dependability in specific did not receive much attention. In the paper, we explore several solutions to increase dependability in fog computing such as fault tolerance techniques, placement policies, middleware, and data management mechanisms aiming to help system designers choose the most appropriate solution.

1. Introduction

The exponential increase in Internet of Things (IoT) technology led to digitizing all aspects of our lives. The number of connected devices is expected to reach 50 billion devices. IoT devices range from household devices to industrial, autonomous transportation, smart cities, and environmental monitoring sensors/actuators. All these devices need to be connected to the internet to meet their prospective. Cloud computing with its services (platform, infrastructure, and software) can be a propitious option for IoT devices.

However, cloud computing cannot handle the massive growth. Contacting cloud data centers is considered to be expensive and rises network bottleneck congestion issues. Furthermore, the delay tolerance of IoT devices varies depending on how critical they are. For example, health care and traffic-controlling IoT devices require lower response times than household IoT devices. At this level, the need for a decentralized computing architecture has emerged. In 2012, Cisco has introduced fog computing as an extension to cloud computing which focuses on bringing data processing geographically closer to the data source (CS, 2015). OpenFog consortium defines fog computing as "a horizontal system-level architecture that distributes computing, storage, control, and networking functions closer to the user along to a cloud-to-thing continuum" (OFC, 2017). This augmentation allows distributing the workload over widespread computing resources to reduce response time and bandwidth consumption resulting in a higher quality of service. Several hierarchies of fog computing can be implemented based on the system requirements, yet the most common one is illustrated in Fig. 1.

As depicted, the lowest layer of the architecture consists of IoT sensors/actuators and mobile devices. The middle layer contains fog devices, which have relatively limited computing resources like gateway servers, routers, switches, access points, or desktop computers. Lastly, the highest level is cloud data centers, which have supercomputing capabilities in terms of processing and storage mediums (OFC, 2017).

Heterogeneity of fog nodes and end devices in fog computing architecture is a major feature that helps to utilize already existing computing resources, yet it complicates the managing process (Bansal and Kumar, 2020). Such a diversified environment must be considered when designing fog environments in terms of computing capabilities, power consumption, and connectivity. Furthermore, the

* Corresponding Author.
Email Address: s.alraddady@latrobe.edu.au (S. Alraddady)
https://doi.org/10.21833/ijaas.2021.04.010
Corresponding author's ORCID profile:
https://orcid.org/0000-0001-6228-9696

decentralized nature of fog computing along with its heterogeneity rise the importance of questioning its dependability in order for business holders and decision-makers to adopt this new computing paradigm.

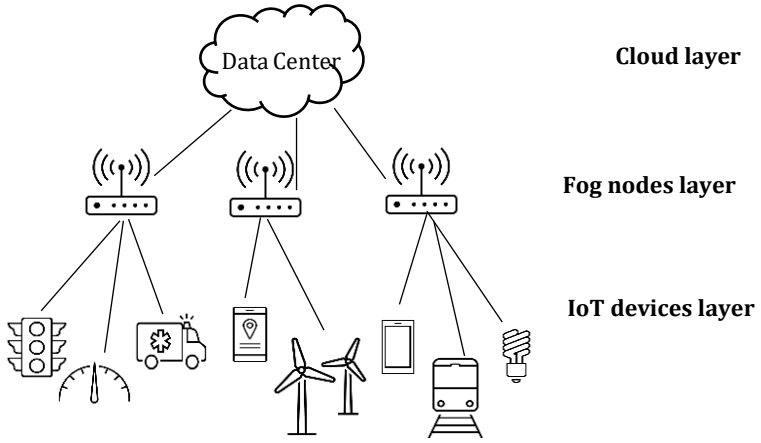

Fig. 1: Architecture of fog computing

An example that clarifies the importance of studying fog applications' dependability can be found in the smart city scenario. Fog computing applications in smart cities can be used to monitor public security in a form of smart cameras for CCTV which generate a huge amount of data that makes it very challenging to be processed and stored. Another area that can benefit from fog applications in smart cities specifically is environmental monitoring in which sensors read and calculate air, lakes, and noise pollution and detect natural disasters (Mohamed et al., 2019). Accordingly, dependability measures vary depending on fog applications.

This paper is structured as follows: Sections 2 discuss relevant research papers. Section 3 discusses the dependability of fog computing and differentiates between availability and reliability as performance metrics, and surveys different mechanisms to improve the dependability of fog computing including fault tolerance techniques, placement policies, middleware, and data management. Section 4 concludes the paper.

2. Related works

Although fog computing is a relatively new computing paradigm, studies focusing on the challenges and issues of fog computing are numerous. Existing studies fall into two categories. The first category contains studies that address general problems that might arise when deploying fog computing; for example, Arivazhagan and Natarajan (2020) pointed up several challenges to be addressed in the fog paradigm including effective security mechanisms related to authentication and denial of service. Reliability, context awareness, user context, deploying fault-tolerant fog nodes, bandwidth management, energy consumption, and data dissemination were also brought out as major issues. Also, Bansal and Kumar (2020) provided a taxonomy for IoT systems that focuses on system architecture, communication, middleware, security, and the relationship between IoT systems and fog computing, and big data. The second category includes studies that investigate a specific aspect of fog computing; for example, Heidari et al. (2020) focused on offloading mechanisms for IoT devices since fog computing systems include IoT devices that have limited computing resources and in need of more powerful computing capabilities to handle the immense volume of data generated. Authors survey a number of offloading mechanisms and offer a parametric comparison to help practitioners and system designers choose the most suitable mechanism. Moreover, Moura et al. (2020) addressed issues of fog computing in the healthcare sector by analyzing the most significant published related work in the past 10 years several challenges were tackled such as interoperability, security, big data, and resources management.

Therefore, the importance of our paper relies on the fact that it focuses on a specific aspect of fog computing which is dependability unlike the first category of existing work. Moreover, enhancing the dependability of fog computing did not receive much attention. We hope this paper provides better insight for system designers when it comes to deciding what is suitable for a fog environment.

3. Dependability in fog computing

Generally, dependability is a broad term that encapsulates the reliability, availability, performability, maintainability, and safety of a system (Johnson, 1989). This paper focuses on the reliability and availability of fog computing in precise.

Reliability in distributed computing systems is a broad term that can be related to system's proper functionality without any service disruption, system's design in terms of how the requirements are met, and the system's ability to tolerate failures and recover from them without undesired results (Ahmed and Wu, 2013). Reliability in fog computing is defined as the continuous functioning of a system in normal or adverse conditions as stated by

OpenFog Consortium (OFC, 2017). Like any distributed computing system, if the system's components are not well-connected or lack orchestration, the system's performance will not satisfy. Given the fact that fog computing is a highly heterogeneous environment, the reliability of fog computing can be considered as a combination of three aspects: Device reliability, connection reliability, and software reliability (Popentiu-Vladicescu and Albeanu, 2017).

Availability is defined as "the probability that the system can complete a predetermined function under a predetermined condition" (Dong et al., 2013). From a business perspective, according to Liu et al. (2010), service availability definition differs from users' perspective to business or service provider. Users define service availability as response time while service provider defines it as how much the provided service contribute to business vitals. However, one of the fundamental service availability measures is Service Level Agreements SLA. These agreements combine the interests of users and service providers to determine and evaluate the provided service. Availability in fog computing is defined as continuous management and orchestration, which is usually measured by up time as stated in OpenFog Consortium (OFC, 2017). Nguyen et al. (2020) highlighted that availability is a metric of quality of service (QoS), and service disruption can lead to undesirable outcomes including business revenue loss, resource damage, or even human loss.

The difference between reliability and availability relies on the time factor. For reliability, the system's performance is calculated for an interval of time. On the other hand, the system's availability is calculated for an instance of time (Johnson, 1989). Improving reliability and availability in fog computing paradigms is still in the early stages and did not receive much attention, yet some remarkable studies have been conducted and will be investigated in this paper.

Ensuring high dependability in fog computing can be achieved via different methods. These methods include fault-tolerant techniques, middleware, context-aware placement policies, and data refining mechanisms. Using fault tolerance techniques mitigates the ramifications of faults in fog nodes and network connectivity, while middleware helps to manage the heterogeneous nature of fog computing to improve QoS. Moreover, context and mobility-aware placement policies increase the availability of fog computing by choosing the most suitable algorithm to process queued requests based on request priority and number of requests to ensure load balancing and enhance overall system performance. Lastly, data refining mechanisms reduce the amount of data sent to the main cloud by identifying and sending meaningful data instead of forwarding raw data which results in higher QoS and better resource utilization. The next sections highlight the challenges and solutions for each of the above methods.

3.1. Fault tolerance techniques

Fault tolerance has emerged since the beginning of distributed computing. This concept is defined as a systems' survival attribute in which it can operate with the existence of faults in any part of it. Fault tolerance techniques have been introduced to minimize fault manifestation in computing systems. These techniques consist of extra hardware, software pieces, or extra time slot to replicate a certain job. This replication is used as a backup in case of failure (Johnson, 1996). Redundancy, watchdog, checkpointing, and retry are the most popular fault tolerance techniques.

Focusing more on enhancing service availability in fog computing, the authors in Grover and Garimella (2018) presented a fog computing architecture that is capable of replicating data at edge nodes. In the architecture, three types of nodes are presented. They are (1) Dew Nodes: Located at the extreme edge of the network and have small computational power. These nodes are suitable for real-time processing. (2) Mist Nodes: processing nodes that can be placed at an institutional level with higher computation power when compared to dews. (3) Fog Nodes: these nodes have storage capacity and are higher in the hierarchy than dew and mist nodes. They can be accessed by ISP and many mist nodes can connect to a fog node. (4) Cloud Server: The highest level of the hierarchy. The novelty of this work relies on data replication at the edge of the network. All data sensed by IoT devices are replicated in a parallel fashion at the same level. Once a shutdown or a failure occurs, the mobile agent starts investigating the incident and its effects and respond accordingly. A mobile agent (MA) is a piece of software that can travel across a network to perform certain tasks on behalf of users (Pham and Karmouch, 1998). Here, MA runs on all dew, mist, and fog devices to share information that is used to communicate with other devices in the network and to fetch priority index in case of failure to be carried out at a different node from the same level or a higher one. Simulation results show that using an intelligent agent as MA enhances the performance of fog computing when compared to a centralized computing environment in terms of CPU consumption and application assignment during faults.

Addressing hardware failures or lost communication with cloud layer issues; Javed et al. (2018) designed three layers of fault-tolerant architecture for IoT applications. The main focus of the proposed architecture is to build a highly available system that can operate even when the connection to the main cloud is lost or any physical harm has been done to edge nodes by replication data at the edge. The authors believe that local data replication ensures data fault tolerance in scenarios like harsh environments or criminal activities occurrence. The three layers in the architecture are (1) Application Isolation layer which is responsible for wrapping processes into independent blocks

using Linux containers, Docker in specific. (2) Data Transport layer providing a publish/subscribe messaging framework for data replication within the cluster, and also support data transport in data pipeline form. (3) Multi-cluster Management layer to monitor the above layers and assign requesting to physical nodes based on fault tolerance requirement and load balancing. The architecture has been applied on a surveillance camera scenario to evaluate the proposed architecture in terms of fault tolerance, in this case, physical damage. The results show that the system is able to tolerate a failure of two nodes out of five nodes. Additionally, data that was replicated within the cluster or at the cloud was reachable even after the damage which alleviates the outcomes of losing two nodes.

Choudhury et al. (2019) presented an event-based service replication scheme for a fog quasi-Adhoc environment to increase system availability. In smart city scenarios, users' mobility throughout the day can provide computing resources at a certain time of the day more than the rest of the day based on users' movements. Choudhury et al. (2019) designed a scheme that can maximize resource utilization during such times. The scheme consists of three modules that function correlatively in order to decide what service needs to be replicated and what node is most suitable to function as a replication agent based on context. The first module is the proactive sensing module designed to sense all events that might require replication. The sensing outcome from this module is called an event. Each event includes a pair of the service that needs replication and source node. All events are forwarded to the next module which is the context computation. This module calculates the logical-physical context to decide the suitability of server nodes. The last module in the proposed scheme is the decentralized computation distribution problem (DCDP) module which is responsible for mapping the services to the replicating nodes in a resource utilizing manner while maintaining the required quality of service. The performance of the scheme has been evaluated and compared to recent algorithms in terms of service availability, response time, and resource utilization. The simulation results of the proposed algorithm showed performance improvement based on the previously mentioned criteria in comparison to the existing algorithms.

Furthermore, Guerrero-Contreras et al. (2017) presented a mobile clouds architecture that focuses on enhancing service availability using dynamic service replication in that activating and hibernating service replication module is based on context. Depending on monitored information, the service replication module selects the most suitable node to operate as a replica for the requesting node. The limitation of the proposed model is its lack of flexibility. New services are not allowed during run time. Furtherly, since this model relies on mobile devices, the mobility nature of the devices degrades the availability of the system due to battery constraints and network connectivity.

Another fault tolerance technology used to enhance reliability, called checkpointing and retry, has been used by Neto et al. (2018). The checkpoint technique performs interval checkpoints which the system can roll back to when a failure occurs. The retry technique restarts request executing from the last checkpoint. In their presented work, Neto et al. (2018) designed an agent-based architecture to predict price changes for virtual machine services with transient servers offered by Amazon AWS supported by machine learning. They used real data provided by Amazon AWS of price changes in 12 months. Based on the results of their experiments involving the architecture, the authors reached an accurate prediction of 94%.

3.2. Placement policies

Placement policies play a major role in improving fog computing's dependability since placement policies are responsible for determining where to process all requests placed by end-users to reach a higher quality of service. Given the fact that context, mobility, and latency are the major factors that affect fog systems' dependability, researchers have proposed several placement policies that are capable of handling changes in these factors to cope with the dynamic nature of fog computing to ensure higher systems' uptime and reasonable recovery time. Maiti et al. (2019) evaluated several algorithms to select the lowest cost fog node which is randomized, greedy, k-median, k-means, and initial centroid finding method. MATLAB is used to evaluate the effectiveness of each method in reducing latency. Also, the relationship between the number of nodes and latency has been observed. Results of the study can be used in designing fog computing environments in order to reach service level agreements.

Designing a mobility-aware fog computing environment for a smart city; Bittencourt et al. (2017) presented a fog computing model which includes context-aware placement policies to enhance the quality of service of fog computing leading to higher availability. Used placement policies are first come first served, delay priority, and concurrent. Users' mobility (or density in a certain area) is the major factor in deciding which policy is more efficient to handle end users' requests. For example, when requests density increases at a certain location during rush hour, fog nodes change placement policy from FCFS to delay-priority. In order to evaluate the model, two fog applications were presented here, viz. the EEG game and VOST surveillance application. These two applications are from different classes (delay priority, real-time) respectively. Simulating the model included running the two applications with the three different placement policies, and it showed that the concurrent strategy resulted in the quality of service degradation which was avoided by the second strategy FCFS. Delay priority strategy, on the other hand, caused the surveillance application higher

latency than the other two strategies while the game application performance was more satisfactory. In conclusion, to achieve the potential of the fog computing paradigm, selecting placement policies must take into consideration users' mobility which requires deep study of the fog environment.

Lera et al. (2018) presented placement police for fog services. Here, availability is measured by the number of services IoT devices received within their time ratio. The study focuses on examining community relationships between fog devices to improve service availability. The proposed strategy consists of two phases using community structured fog devices. The first phase maps the application to fog devices within the same community, and the second phase is responsible for services placement by prioritizing interrelated services. Lera et al. (2018) used transitive closures to partition the application and combination of complex network communities for devices partitioning. The result of the proposed work after testing two scenarios was that the presented policy outperformed the integer linear programming approach in terms of response time.

Additionally, Mahmud et al. (2019) proposed a placement policy for fog computing environments to enhance users' quality of experience (QoE) by improving resource availability. Fog applications in this environment have two modules which are the client module and the main application module. The used architecture for this specific placement policy consists of two types of fog nodes which are fog gateway nodes (FGNS) and fog computational nodes (FCNS), and they differ in capabilities since each one of them has a different task to accomplish. FGNS can be handheld devices or cable modems and is responsible for mapping the main application module based on the received user's expectations and FCNS's resources index using fuzzy logic models. This approach prioritizes users' expectations while maintaining better resource utilization to improve users' QoE. Evaluating the proposed policy, QoE's measured metrics were service availability, service processing time, and resource affordability was calculated during simulating the policy using iFogsim. Simulation results showed that 92% of applications received a higher processing time reduction ratio in terms of resource gain and network relaxation ratio.

3.3. Middleware

Middleware is a program that enables the communication between entities, which can be software or hardware elements, in distributed computing environments. It simplifies combining, developing, and executing applications without complexity hassles. Since fog computing is not only a distributed environment but also a very heterogeneous one, middleware can be very advantageous in connecting such diversified components. Mohamed et al. (2017) presented a service-oriented middleware (SmartCityWare) for fog computing in smart cities to enhance availability in fog environments. It is designed as a utilization tool to support services in smart cities. All functions in the middleware are treated as services either core services or environmental ones. Core services can be broker, security, or location-aware services, which are services related to management. On the other hand, environmental services are all the services provided by any component (Cloud, fog, and IoT). The main function of the middleware is to facilitate communication smoothly between all aspects and provide an interface to support smart city applications. An example of these applications was introduced to explain the mechanism of the middleware and evaluate the middleware performance in terms of response time and service lookup time. What makes SmartCityWare different from other middleware platforms is the fact that it considers all components as services for smooth services integration which results in unlimited opportunities for development.

Furthermore, one of the challenges that need to be addressed is the amount of data produced by IoT devices. Clemente et al. (2017) addressed the issue of raw data and their effect in downgrading the quality of service in fog computing. They designed a distributed cooperative data analytics middleware (DCDA) to mitigate bandwidth limitations and latency caused by IoT raw data and reach a higher QoS rate. The main idea of the middleware is to maintain data processing and analysis at the edge level to reduce latency. It manages three different levels of data which are: Operational (low/edge level), historical (intermediate/fog level), and filtered data (high/cloud level). Each node in the proposed scheme has some computing capabilities, a library manager, and an adapter to select the correct algorithm from the functions' library and a middleware visualizer that handles results monitoring. Two case studies that require real-time processing were tested in order to evaluate the middleware. Simulation results proved that analyzing data at edge level significantly reduces latency and increases scalability and robustness since the proposed middleware reduced bandwidth cost. Moreover, fault tolerance of the middleware was investigated by simulating the failure of links and nodes, and no considerable impact was witnessed. In conclusion, the proposed middleware proved that processing and analyzing data at the edge level significantly reduces latency by eliminating sending raw data to cloud data centers and decrease bandwidth cost.

3.4. Data refining mechanisms

Data processing, storing and analyzing in fog computing play a major role in the overall performance since IoT devices can produce a massive amount of data. Refining data before processing is one technique to increase dependability in fog computing specially in semi-critical environment where some requests are non-

delay tolerant compared to other requests in the same environment. When data is filtered, higher availability is ensured since fog nodes are not being occupied by meaningless data processing. Accordingly, Wang et al. (2020) presented a framework that is designed for IoT healthcare systems for elderly people. Framework's functionality consists of three aspects: Fault-tolerant data transmission, self-adapting filtering, and data load reduction. Architecturally, fog nodes in the framework fall into two categories which are storage nodes and processing nodes. Storage nodes store data temporarily to be sent later to the cloud data center and sent to processing nodes upon request. For reliable data transmission, storage nodes have more than one connection to processing nodes to track lost data packets through broadcasting and flooding mechanisms. Filtering is the next step after ensuring that data are complete. Predefined requests have been used to classify requests. Once the request is received, category checking is executed to identify health-related requests to be prioritized. Further, the risk assessor function is used to determine the danger level of placed requests that will send an alarm once a threshold has been hit which means an elderly person is in danger. For allocating requests' queue, a simplified version of the reduced variable network search mechanism is implemented. This mechanism is useful since it provides processers the capability to prioritize more valuable data. Based on the results of simulation on the framework, the authors concluded that using self-adapting and fault-tolerant mechanisms ameliorate data transmission in fog computing.

Managing data storage is another technique presented by Steffenel (2018). Enforcing control on where to store data and which aspect of a fog computing environment is more efficient to handle it can improve reliability, according to the author. Here, data locality and locality-aware scheduling mechanisms are presented. Data storage is provided through modified distributed hash tables to include location keys to be used in task allocation. This step contributes to increasing the probability of processing data at a nearby node that is closer to the data source or meets resource requirements. Additionally, the location key is used by the scheduler in task allocations and prioritizing tasks that need local data. Also, to reduce the overhead caused by storing data on fog nodes and to ensure fault tolerance in the proposed framework, data storing mapping can be used to store data in nearby nodes. Reading and writing performances have been evaluated, and the author concluded that the proposed mechanisms can be useful for intensive data fog applications.

4. Conclusion

As IoT devices increase nowadays, a decentralized computing environment is required to manage the massive growth of requests. Fog computing is considered the underpinning for IoT technology to reach its potential. The heterogeneity and distributed nature of this new computing paradigm rises the issue of investigating its availability. Managing and orchestrating fog computing aspects are necessities to reach a satisfactory quality of service. Research on fog computing availability is surveyed to provide a better insight into the challenges and solutions with respect to fog computing dependability. Based on the current literature, we conclude with four methods to deal with the challenges discussed in the paper: fault tolerance techniques, placement policies, middleware, and data refining mechanisms.

Compliance with ethical standards

Conflict of interest

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

References

Ahmed W and Wu YW (2013). A survey on reliability in distributed systems. Journal of Computer and System Sciences, 79(8): 1243-1255. https://doi.org/10.1016/j.jcss.2013.02.006

Arivazhagan C and Natarajan V (2020). A survey on fog computing paradigms, challenges and opportunities in IoT. In the International Conference on Communication and Signal Processing, IEEE, Chennai, India: 0385-0389. https://doi.org/10.1109/ICCSP48568.2020.9182229

Bansal S and Kumar D (2020). IoT ecosystem: A survey on devices, gateways, operating systems, middleware and communication. International Journal of Wireless Information Networks, 27: 340-364. https://doi.org/10.1007/s10776-020-00483-7

Bittencourt LF, Diaz-Montes J, Buyya R, Rana OF, and Parashar M (2017). Mobility-aware application scheduling in fog computing. IEEE Cloud Computing, 4(2): 26-35. https://doi.org/10.1109/MCC.2017.27

Choudhury B, Choudhury S, and Dutta A (2019). A proactive context-aware service replication scheme for Adhoc IoT scenarios. IEEE Transactions on Network and Service Management, 16(4): 1797-1811. https://doi.org/10.1109/TNSM.2019.2928698

Clemente J, Valero M, Mohammadpour J, Li X, and Song W (2017). Fog computing middleware for distributed cooperative data analytics. In 2017 IEEE Fog World Congress, IEEE, Santa Clara, USA: 1-6. https://doi.org/10.1109/FWC.2017.8368520

CS (2015). Fog computing and the internet of things: Extend the cloud to where the things are. Cisco Systems, San Jose, USA.

Dong WE, Nan W, and Xu L (2013). QoS-oriented monitoring model of cloud computing resources availability. In the International Conference on Computational and Information Sciences, IEEE, Shiyang, China: 1537-1540. https://doi.org/10.1109/ICCIS.2013.404 **PMid:23614461**

Grover J and Garimella RM (2018). Reliable and fault-tolerant IoT-edge architecture. In the IEEE Sensors, IEEE, New Delhi, India: 1-4. https://doi.org/10.1109/ICSENS.2018.8589624

Guerrero-Contreras G, Garrido JL, Balderas-Diaz S, and Rodriguez-Dominguez C (2017). A context-aware architecture supporting service availability in mobile cloud computing. IEEE Transactions on Services Computing, 10(6): 956-968. https://doi.org/10.1109/TSC.2016.2540629

Heidari A, Jabraeil JMA, Jafari NN, and Akbarpour S (2020). Internet of Things offloading: Ongoing issues, opportunities, and future challenges. International Journal of Communication Systems, 33(14): e4474. https://doi.org/10.1002/dac.4474

Javed A, Heljanko K, Buda A, and Främling K (2018). Cefiot: A fault-tolerant IoT architecture for edge and cloud. In the IEEE 4th World Forum on Internet of Things, IEEE, Singapore, Singapore: 813-818. https://doi.org/10.1109/WF-IoT.2018.8355149

Johnson BW (1989). Design and analysis of fault-tolerant systems for industrial applications. In: Görke W and Sörensen H (Eds.), Fehlertolerierende rechensysteme/fault-tolerant computing systems: 57-73. Springer, Berlin, Germany. https://doi.org/10.1007/978-3-642-75002-1_5

Johnson BW (1996). An introduction to the design and analysis of fault-tolerant systems. In: Pradhan DK (Ed.), Fault-tolerant computer system design: 1-108. Prentice-Hall, Inc., Upper Saddle River, USA.

Lera I, Guerrero C, and Juiz C (2018). Availability-aware service placement policy in fog computing based on graph partitions. IEEE Internet of Things Journal, 6(2): 3641-3651. https://doi.org/10.1109/JIOT.2018.2889511

Liu H, Lin Y, Chen P, Jin L, and Ding F (2010). A practical availability risk assessment framework in ITIL. In the 5th IEEE International Symposium on Service Oriented System Engineering, IEEE. Nanjing, China: 286-290. https://doi.org/10.1109/SOSE.2010.38

Mahmud R, Srirama SN, Ramamohanarao K, and Buyya R (2019). Quality of Experience (QoE)-aware placement of applications in Fog computing environments. Journal of Parallel and Distributed Computing, 132: 190-203. https://doi.org/10.1016/j.jpdc.2018.03.004

Maiti P, Apat HK, Sahoo B, and Turuk AK (2019). An effective approach of latency-aware fog smart gateways deployment for IoT services. Internet of Things, 8: 100091. https://doi.org/10.1016/j.iot.2019.100091

Mohamed N, Al-Jaroodi J, and Jawhar I (2019). Towards fault tolerant fog computing for IoT-based smart city applications. In the IEEE 9th Annual Computing and Communication Workshop and Conference, IEEE, Las Vegas, USA: 0752-0757. https://doi.org/10.1109/CCWC.2019.8666447

Mohamed N, Al-Jaroodi J, Lazarova-Molnar S, Jawhar I, and Mahmoud S (2017). A service-oriented middleware for cloud of things and fog computing supporting smart city applications. In the IEEE Smart World, Ubiquitous Intelligence and Computing, Advanced and Trusted Computed, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People and Smart City Innovation, IEEE, San Francisco, USA: 1-7. https://doi.org/10.1109/UIC-ATC.2017.8397564

Moura CHJ, da Costa CA, da Rosa Righi R, and Antunes RS (2020). Fog computing in health: A systematic literature review. Health and Technology, 10: 1025–1044. https://doi.org/10.1007/s12553-020-00431-8

Neto AJP, Pianto DM, and Ralha CG (2018). A fault-tolerant agent-based architecture for transient servers in fog computing. In the 30th International Symposium on Computer Architecture and High Performance Computing, IEEE, Lyon, France, France: 282-289. https://doi.org/10.1109/CAHPC.2018.8645859

Nguyen TA, Min D, and Choi E (2020). A hierarchical modeling and analysis framework for availability and security quantification of IoT infrastructures. Electronics, 9(1): 155-184. https://doi.org/10.3390/electronics9010155

OFC (2017). Openfog reference architecture for fog computing. OpenFog Consortium, Fremont, USA.

Pham VA and Karmouch A (1998). Mobile software agents: An overview. IEEE Communications Magazine, 36(7): 26-37. https://doi.org/10.1109/35.689628

Popentiu-Vladicescu F and Albeanu G (2017). Software reliability in the fog computing. In the International Conference on Innovations in Electrical Engineering and Computational Technologies, IEEE, Karachi, Pakistan: 1-4. https://doi.org/10.1109/ICIEECT.2017.7916578

Steffenel LA (2018). Improving the performance of fog computing through the use of data locality. In the 30th International Symposium on Computer Architecture and High Performance Computing, IEEE. Lyon, France: 217-224. https://doi.org/10.1109/CAHPC.2018.8645879

Wang K, Shao Y, Xie L, Wu J, and Guo S (2020). Adaptive and fault-tolerant data processing in healthcare IoT based on fog computing. IEEE Transactions on Network Science and Engineering, 7(1): 263-273. https://doi.org/10.1109/TNSE.2018.2859307