Towards Fault Tolerant Fog Computing for IoT-Based Smart City Applications

Nader Mohamed
Middleware Technologies Lab.
Pittsburgh, PA
USA
nader@middleware-tech.net

Jameela Al-Jaroodi
Department of Engineering
Robert Morris University
Moon Township, PA, USA
aljaroodi@rmu.edu

Imad Jawhar
Faculty of Engineering
Al Maaref University
Beirut, Lebanon
imad@midcomp.net

***Abstract*—Fog computing can provide many services to support IoT-based smart city applications. Fog computing usually consists of multiple nodes that are distributed across a smart city to enable IoT-based smart city applications such as intelligent transportation, smart energy, smart water, smart health, smart infrastructure monitoring, and smart environmental monitoring. The Fog platform will allow executing services geographically close to the IoT-based smart city applications to provide low latency, location awareness, mobility, streaming, management, and real-time support. One of the main issues with this support is the reliability and fault tolerance of the fog platform. This paper discusses the issues of reliability and fault tolerance for fog platforms supporting IoT-based smart city applications. It investigates different considerations to achieve a good degree of fault-tolerance for fog computing supporting smart city applications. The paper also proposes fault tolerance middleware services for fog computing to help solve reliability and fault tolerance issues. These services can provide a more reliable environment to operate IoT-based smart city applications.**

Keywords- Smart City, Fog Computing, Cloud Computing, Fault Tolerance, Middleware, Internet of Things

I. Introduction

Urban population world-wide is on the rise; according to a United Nations study, it has grown from 746 million in 1950 to 3.9 billion in 2014 [1]. This number is estimated to increase by 66 percent to more than 6 billion by 2050. Consequently, many cities around the world are increasing in population while they have very limited or inadequate resources such as water, energy, transportation, and unoccupied land. This will lead to many challenges for the administrations of these growing cities to continue offering and managing the required resources and services to provide a good quality of life for their residents. One possible solution is to utilize new information and communication technologies (ICT) to convert from traditional to smart cities managed by intelligent services that can help optimize operations and resources utilization and intelligently manage all aspects of city life.

The Internet of Things (IoT) and other advanced enabling technologies such as cloud and fog computing can effectively contribute to the emerging concept of smart cities. They offer opportunities to develop and deploy a wide array of applications that can effectively and intelligently manage smart city resources. Integrating IoT with fog and cloud computing can provide an advanced platform for supporting smart city applications [2][3]. Using IoT facilitates the integration of a smart city's physical objects in a city-wide network. Additionally, loud computing provides scalable and cost-effective computation, data storage, and advanced software services to support smart city applications, yet at the cost of time delays between the IoT and the cloud. Fog computing can overcome some of the cloud computing limitations by providing services that operate on fog nodes close to the IoT and the operational locations of the applications, providing support for low latency, location awareness, mobility, streaming and real-time requirements.

An integrated IoT-fog-cloud platform can be used to effectively operate many smart city applications [2][4] and many smart cyber-physical systems in general [5]. In smart cities, this integrated platform can be used to effectively operate smart city applications such as smart energy, smart water, smart transportation, smart security, smart infrastructure monitoring, and smart environment monitoring in smart cities. However, this platform should be reliable to realize its benefits. These applications cannot provide reliable support for smart cities if the underlaying platform is not reliable [6]. While the reliability of cloud computing and IoT were investigated [7][8] and several middleware-based fault-tolerance solutions were proposed for both areas [9][10][11], the reliability and fault tolerance of fog computing is not yet adequately investigated. As the fog computing layer represents the middle layer that connects the IoT with the cloud, it is extremely important for fog computing to be reliable and fault-tolerant. This paper investigates and discusses the reliability issues in fog computing supporting IoT-based smart city applications (IoTSCA). In addition, it identifies different considerations to incorporate fault-tolerance mechanisms to support reliable IoTSCA.

The fault model we will consider in this paper is based on a complete failure of some fog nodes due to reasons such as hardware malfunctions, connectivity issues, power loss, or physical/cyber attacks. This failure should be rapidly discovered and recovered by other fog nodes to maintain good operations for the supported smart city applications. For example, consider a smart traffic light application that handles multiple intersections in a downtown location. Here, there is a fog node with services to handle each intersection, a sudden failure of any of these fog nodes may disrupt the overall operations and cause traffic problems. Therefore, such failure must be rapidly discovered and rectified by other fog nodes in the area to continue the healthy operations of the traffic lights at these intersections.

The paper is organized as follows. Section II provides background information about IoT-based smart city applications and fog computing support. Section III discusses a service-oriented model for IoT-based smart city applications. Fog services types are discussed in Section IV. Section V discusses fault-tolerant fog computing considerations. Some proposed fault-tolerant fog computing services are discussed in Section VI while Section VII concludes the paper.

II. IoT-Based Smart City Applications and fog supports

IoT advancements and its integration into various aspects in urban developments have led to greater steps towards smart cities. Smartness is achieved when we have complete systems that can monitor, analyze and react intelligently to adjust the environment and systems for optimal results. IoT devices provide the capabilities for monitoring and executing responses, while additional ICT components support them to collect, organize, and analyze data to make decisions either on the cloud, fog nodes, or locally at the operational sites.

Applications offering smart services for cities extend to cover all aspects of city life including smart energy leveraging efficient use and management of energy sources and optimizing consumption and greenhouse gases emissions. Other examples are smart water networks; smart transportation systems; embedded systems for monitoring and managing infrastructures and their health; smart environmental monitoring systems and enhanced smart safety and security applications for the residents and infrastructures. Table 1 lists some examples of these applications and summarizes how IoT, fog and cloud computing can support them.

Integrating IoT with fog and cloud computing, as shown in Fig. 1, provides the basis for smart applications serving city needs. In such integration, interaction between the IoT and other components is necessary and can happen in two directions. When the IoT is monitoring the environment, the devices will relay data upward to the fog nodes, cloud nodes or both. The cloud and fog nodes also need to pass data or execute functions available on the IoT devices, thus initiating a downward stream. As fog nodes usually reside between the IoT devices and the cloud, they have an important role in the integration supporting the smart city applications. As a result, failures in fog nodes, could have a strong negative impact on these applications and the environment they serve.

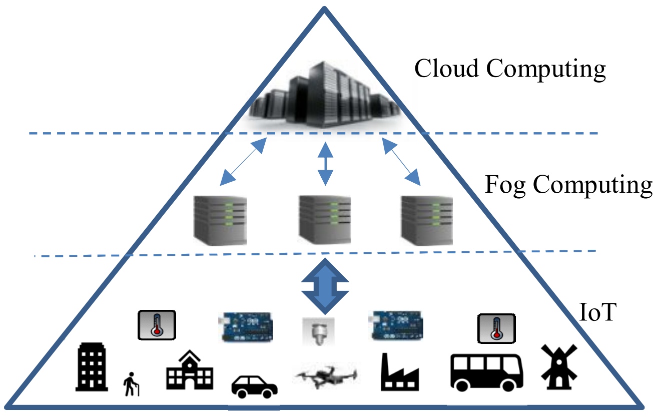

Fig. 1. Integrated IoT, fog, and cloud layered architecture.

The fog nodes are usually fixed nodes provided through dedicated computers or network routers. However, it is possible to have mobile fog nodes. One example of mobile fog nodes is UAVFog [12]. UAVFog is UAV-based fog computing. UAVFog offers fog computing capabilities onboard one or more UAVs that can dynamically provision IoT services and applications at several places. UAVFog is an extremely mobile and supple fog platform that can be quickly moved to various areas to provision certain smart city services and applications as needed.

Fog computing nodes can be deployed to provide services for IoTSCA using either a flat or a hierarchical structure [13] depending on the type of the applications to be deployed. In a flat structure, all fog nodes are organized in one level and they provide similar types of services. In contrast, in a hierarchical structure, the fog nodes are organized in multiple levels where each level provides different types of services from the other level. The lower-level fog nodes are located close to the IoT devices while upper-level fog nodes provide other higher-level services for its connected lower-level fog nodes. In addition, the top-level fog nodes are connected to the cloud.

As in any type of large-scale distributed application, there are risks of failure in any component of an IoTSCA. IoT components and devices may fail, yet it is usually easy to detect and rectify such failures and provide enough redundancy to avoid critical problems. Cloud services may fail as well; however, cloud service providers, offer many proven models to mitigate this risk and respond to it effectively with minimal effects on the clients' applications using them [7][14]. Similarly, fog nodes and fog-based services are also prone to failures that could drastically affect the applications. However, there is not much work investigating these types of failures or offering workable solutions to them. As shown in Table 2, failures in fog nodes providing critical services to the IoTSCA could lead to many problems due to the loss of connectivity to one or more fog nodes or the absence of critical services due to different types of faults. As a result, it is essential to find solutions to detect and respond to such failures, at the same time, minimize the effects on the applications.

Table 1. Examples of IoTSCA that can benefit from an Integrated IoT-Fog-Cloud platform.

Smart City Application	**Sub-applications**	**IoT Components Functions**	**Fog Nodes Functions**	**Cloud Functions**
Smart energy systems	• Smart grid • Smart buildings • Renewable energy plants • Wind farms • Hydropower plants	Link different energy-related sensors, smart meters, and actuators for different energy systems in a network that can be linked with a city network.	Provide local controls and real-time responses for energy systems, distribution units, and consumer locations. They also enable smooth integration of different energy systems.	Collects, filters, and stores energy information. Supports decision making for utilizing smart grids and renewable energy features based on collected and analyzed data for consumers' needs and renewable energy productions.
Smart water systems	• Leakage and other failure detection • Water leakage reduction • Water quality monitoring • Smart irrigation • Smart water supply networks	Link water network-related devices such as smart water meters, water quality sensors, and water control actuators in a network that can be connected to other networks.	Provide better and faster local monitoring and controls for smart water networks. They also offer real-time monitoring for faults and leakages and support repair and maintenance operations.	Collects, stores, and utilizes smart water networks data to enhance the water networks, production, and quality and to reduce water losses. It also supports long term planning and management.
Smart transportation systems	• Route planning and congestion avoidance • Smart traffic lights • Smart parking services • Accident avoidance • UAV Taxis • Autonomous vehicles	Link different road sensors, traffic light controllers, vehicular on-board units (OBUs) and other mobile and fixed devices in a network. This network can be linked with other city networks.	As Road Side Units (RSUs) or other computerized units provide low-cost relays among vehicles', roads' and parks' sensors; traffic lights; and the cloud. They provide fast response and control services.	Collects, filters, and stores traffic information. It helps in coordinating city traffic and parking optimization. It also helps in planning for enhancing traffic systems.
Smart city structural health monitoring and management systems	Structural health monitoring and management for: • Bridges and Tunnels • Large public buildings • Train and subway rails • Oil and gas pipelines	Link different structure health monitoring sensors in a network that can be connected to other city networks.	Help reduce data traffic between the sensors monitoring the structures and their main control stations. In addition, they provide fast safety controls for some applications.	Collects, filters, and stores structural health data. It helps analyze collected data to enhance the maintenance processes and improve the health of the city's structures.
Smart environmental monitoring systems	• Air quality monitoring • Noise monitoring • Rivers, lakes and oceans monitoring • Smart natural disasters monitoring and prediction	Link environmental monitoring sensors and control devices in a network. This network is linked with other city networks.	Help enhance environmental monitoring processes by providing smart environmental monitoring and analysis closer to the monitored sites.	Provides processes to collectively analyze the city's environmental and health status. These processes can offer suggestions for environmental solutions.
Smart public safety and security	• Crowd control for large events (sports games, concerts, parades, and outdoor celebrations) • Crime watch and alerts • Large-scale emergency response services (floods, earthquakes, terrorist attacks, and volcanoes)	Link different security and sensor related sensors, cameras, and security monitoring UAV in a network.	Help reduce the communication traffic between the events' locations, the main security monitoring stations, and the security and rescue teams.	Provides a powerful platform for analyzing the collected data about the current situation to help in providing optimized action plans for better controls and emergency relief. It also offers further analysis to help improve procedures for future events.

Table 2. Risks of fog failures to IoTSCA.

Smart City Application	**Possible Risks of Fog Failures**
Smart energy	Energy waste, and possible hazardous situations like fires and power overloads.
Smart water	Water waste, failures in water quality control procedures.
Smart transportation	Vehicular accidents, more traffic congestion, and high transportation delays.
Smart city structure health monitoring	Major or minor city infrastructure failures, costly maintenance and repair operations.
Smart environmental monitoring	Wrong environmental monitoring data, inaccurate responses, and negative impact on human health and environment.
Smart public safety and security	Injuries and possible losses of life, and losses and damages of city infrastructures.

III. A SERVICE-ORIENTED MODEL FOR IOT-BASED SMART CITY APPLICATIONS

The key to successful design and implementation of IoTSCA is effective and efficient integration between IoT components, fog nodes and the cloud. Several approaches and programming paradigms can be used to achieve this goal. One of these is the programming paradigm based on the service-oriented architecture (SOA). SOA provides advanced and flexible features for smart city applications development and deployment in such integrated manner. SOA uses "services" as the unit of computation and/or communication. A collection of these service can be used to design, implement and deploy integrated IoTSCA. One of the main advantages of this paradigm is the cost-effective delivery of standalone or composite IoTSCA that can "integrate from the inside-out." SOA offers many advantages like flexibility, interoperability, automatic integration capability, reusability, modularity, and openness [15][16]. It also lends itself naturally to the distributed nature inherent in the large geographic distribution of a smart city's infrastructure and resources.

Services can be used to cleanly represent the different components of the IoTSCA. IoT devices, smart devices, control components and any other components used can be represented by one or more services. As such access and integration become more uniform through the standard service interfaces. In addition, software processes and work procedures are also represented as services residing on the smart devices, fog nodes or the cloud. As such, they become available and usable through the same set of interfaces.

The SOA approach allows for introducing support features and functions that can benefit IoTSCA. Generic services to support requirements like reliable communication, enhanced data flow, advanced analytics, fault tolerance, load balancing, security, and communication management to name a few can be designed and implemented to support different applications. To facilitate such paradigm, we developed SmartCityWare [2], a service-oriented middleware (SOM) designed to develop and operate distributed smart city services. SmartCityWare provides base services to integrates the different technologies needed for smart cities such as IoT and fog and cloud computing. SmartCityWare views all resources including the IoT components as services that can be used by and use other services in the IoTSCA. SmartCityWare also provides core services such as broker, invocation, location-based, and security services. Here we extend SmartCityWare to include services to create a fault-tolerant fog computing platform supporting IoTSCA.

IV. Fog Services Types

Fault-tolerant fog computing can be performed at the level of fog services. When a fog node where a fog service that supports IoTSCA fails, it can be replaced by a similar service available on another healthy fog node located close enough such that the application can continue its operations without problems. Different fault-tolerance mechanisms and optimizations can be achieved for a fog service based on the type of the service. Some services require more considerations than others to achieve fault tolerance. Therefore, it is important to first discuss the different types of fog services to leverage the understanding of different possible fault-tolerance issues and mechanisms for these types.

A. Stateless Services Vs. Stateful Services

There is a major difference between stateful and stateless fog services in maintaining or not maintain state data. A stateful fog service maintains state data that can be updated with each service invocation and the operations and outcomes of this service can be affacted with the current state data. Examples of stateful fog services are a service that coordinates and synchronizes several local IoT devices and a service that provides average measurements of a sensor within a defined period. In both cases, the services need to keep information that propagate from one invocation to the next and cannot operate correctly without it. This type of service must be replicated with all its current state data and the replica must continuously updated with any changes in the state data, which makes it difficult to accurately maintain at all times. In contrast, a stateless fog service is a service that performs a specific task upon its invocation without requiring or updating any state data and both its operations and outcomes are not affected by the state data. This service can be a computation service that performs a specific calculation on a provided data, a service to retrieve the current measurement of a sensor and pass it on to another service, or a service to forward a message to a server. Generally, a stateless service can be easily replicated on another node as it does not maintain any state data.

B. Non-Real-Time Services Vs. Real-Time Services

A real-time fog service is a service that needs to take certain actions and/or offer instantaneous responses within a certain time frame. These actions and responses become useless if they are not completed within the specified time. Therefore, the operations of the application using the service may be negatively affected if the actions or responses are not delivered in time. This type of service can be better supported by fog computing than cloud computing. The availability of fog nodes close to the IoT devices can provide better control facilities being closer to the source/destination of the data and actions. This allows real-time services to perform many tasks to support IoTSCA correctly and quickly. On the other hand, fog computing can still support non-real-time services that do not have such time constraints to perform their tasks. It is easier to handle fault tolerance with non-real-time fog services compared to real-time fog services. This is mainly due to the time constraints as non-real-time services can be offloaded to any available fog node or even cloud node with minimal impact. However, a real-time service must be relocated to another fog node that can still satisfy the required time constraints. Therefore, it is important to identify the suitable backup fog nodes that are close enough and have the required capabilities to perform the same service.

C. Single-Level Services Vs. Multiple-Level Services

A single-level service is a service that completes its operations upon its invocation within one fog node and without involving any other services available at other levels such as the cloud, upper-level fog nodes, lower-level fog nodes, or IoT devices. However, a single-level service can invoke other single-level services available within the same fog node. In contrast, a multiple-level service is a service that needs to use other services available on other nodes at different levels to complete its operations. These other nodes can be the cloud, other fog nodes at any level, and IoT devices. The main difference between a single-level and a multiple-level service in the terms of fault-tolerance is that a single-level service can be easily handled on another fog node with similar capabilities. However, a multiple-level service requires ensuring connectivity to all other levels it needs to communicate with to complete its task. For any single-level service in the case of node failure, it is only important to replicate or start the service with other involved local services on another fog node that can be reached by the same clients to complete the operations. While for a multiple-level service, the replica must be made and or started on another fog node that is reachable by the same clients and nodes/levels where the necessary services are located. Therefore, it is important to create the correct connections to any other IoT devices, fog nodes and cloud services needed.

D. Different Types of Fog Services

Based on the above discussion, there are 8 type of fog services for IoTSCA. These services types are shown in Table 3 and they exhibit different levels of difficulty to handle fault tolerance. For example, fault tolerance for single-level, non-real-time, and stateless services is the easiest as there are limited constraints and replication can be done easily. On the other hand, multiple-level, real-time, and stateful services are the most complex as they require satisfying multiple constraints and continuous monitoring for replication.

Table 3. Combinations of different types of services.

Number	Service Type
1	Single-level, non-real-time, and stateless services
2	Single-level, non-real-time, and stateful services
3	Single-level, real-time, and stateless services
4	Single-level, real-time, and stateful services
5	Multiple-level, non-real-time, and stateless services
6	Multiple-level, non-real-time, and stateful services
7	Multiple-level, real-time, and stateless services
8	Multiple-level, real-time, and stateful services

V. FAULT-TOLERANT FOG COMPUTING CONSIDERATIONS

There are several considerations to study to achieve fault-tolerance for fog computing supporting IoTSCA:

1. We need to configure all IoT devices to communicate directly or indirectly with at least two close by fog nodes. This is necessary to continue providing all or some important fog services even with some fog node failures. For example, for smart traffic lights all IoT devices on any of the lights should be able to communicate directly or indirectly with at least two fog nodes. Each IoT device will use one of these fog nodes as its primary fog node while the other is considered a backup node.
2. Better fault-tolerance can be achieved if we have overlap in coverage of fog nodes across the IoT distribution areas. It is better to allocate fog nodes to be able to support multiple IoT areas. This is valid not only for a flat-structure fog computing, but also for all levels in a hierarchical structure fog computing.
3. Failure of any fog node can suddenly occur. Although this failure can be discovered by one of the connected IoT devices or a cloud service that uses a service available in the fog node, this can take time. Another way is to use the watch-dog technique from the cloud. However, due to possible high communication delays between the cloud and the fog nodes, discovering failures may take an unacceptable amount of time that exceeds the time constrains of real-time services. Instead, it is more feasible to create a structure that allows each fog node to be monitored by other nearby fog nodes, thus localizing the process and providing faster fault discovery.
4. Replacing a faulty fog node with another to provide the required services is generally based on three conditions:
   - Resources: the replacement node has enough resources in terms of computation, storage, memory, and connectivity to handle the additional load and communication requirements.
   - Accessibility: The replacement node is accessible by all its clients including the IoT devices, fog clients and cloud clients. Furthermore, accessibility should be maintained for multiple-level services.
   - Timing: the replacement node can maintain the timing conditions for real-time services.
5. Backup alternatives are dependent on the types of services used (see Table 3). For example, creating a backup for a stateless non-real-time, single or multiple level service only requires ensuring having the right connections to the backup fog node and a single image with no updates, unless the service is changed. It may also be possible to create the backup on the cloud. However, when considering a real-time stateful service, it becomes important to create a backup on a node that can satisfy the real-time requirements and create a periodic update mechanism for the state data.
6. Replicating current state data for non-real-time stateful fog services can be done using the cloud, as it has enough resources and time is not an issue. However, replicating current sate data for real-time stateful services should be done within the neighboring fog nodes. This will help maintain the timing requirements of real-time services.

VI. FAULT TOLERANT FOG COMPUTING SERVICES

Several services can be provided to support fault tolerant fog computing. Here we identify some of these services supporting IoTSCA. The selected services are based on the different types of services and different considerations discussed earlier. These services can be added to SmartCityWare to include fault-tolerance features for IoTSCA.

- Watch-Dog: to monitor a specific fog node by another fog node.
- ServiceReplication: to replicate a stateful fog service on another fog node.
- ReplaceMainBroker: to change the broker an IoT device uses to another broker on the replacement fog node.
- FindFogNodes: to find a list of a node's neighboring fog nodes.

- FindIntermediateNodes: to find a list of fog nodes between two other specified fog nodes (for multi-level services).
- FindCommunicationTime: to find the average communication time between two fog nodes.
- FindResourceStatus: to find the resources status of a specific fog node.

The above services and several others can be used as building blocks to build fault-tolerant fog computing features to support reliable IoTSCA. In addition, they can be used to build an advanced fault-tolerant fog computing environment.

VII. Conclusion

Using SOA helps support the introduction of fault tolerance for fog computing supporting IoTSCA. Specific services to monitor other services and devices, track operations and help discover failures quickly can be added and activated. To achieve this, we need to identify the different types of services: stateless vs. stateful; real-time vs. delay-tolerant; and single level vs. multiple level. This creates the basis to recognize the different considerations to take into account when designing and using fault-tolerance mechanisms for fog computing. Fault discovery, backup and update mechanisms; services reallocation; and connectivity and replication are some of these considerations. Services to monitor fog nodes, find resources status, adjust broker operations are examples for fault tolerance services that can be implemented within SmartCityWare to provide fault-tolerant fog computing supporting IoTSCA.

References

[1] United Nations, "World Urbanization Prospects - The 2014 Revision," ISBN 978-92-1-151517-6, United Nations, 2014.

[2] N. Mohamed, J. Al-Jaroodi, I. Jawhar, S. Lazarova-Molnar, and S. Mahmoud, "SmartCityWare: A Service-Oriented Middleware for Cloud and Fog Enabled Smart City Services," in IEEE Access, Vol 5, pp. 17576 – 17588, Dec. 2017.

[3] N. Mohamed, J. Al-Jaroodi, S. Lazarova-Molnar, I. Jawhar, and S. Mahmoud, "A Service-Oriented Middleware for Cloud of Things and Fog Computing Supporting Smart City Applications," in proc. 2017 IEEE Int'l Conference on Smart City Innovations (IEEE SCI 2017), pp. 1152-1158, 2017.

[4] C. Perera, Y. Qin, J.C. Estrella, S. Reiff-Marganiec, and A.V. Vasilakos, "Fog computing for sustainable smart cities: A survey," ACM Computing Surveys (CSUR), 50(3), p.32, 2017.

[5] J. Al-Jaroodi and N. Mohamed, "PsCPS: A Distributed Platform for Cloud and Fog Integrated Smart Cyber-Physical Systems," in IEEE Access, Special Section on Cyber-Physical Systems, IEEE, Vol. 6, pp. 41432-41449, 2018.

[6] E.Z. Tragos, et al. "Enabling reliable and secure IoT-based smart city applications," In IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), pp. 111-116, 2014.

[7] R. Jhawar, V. Piuri, and M. Santambrogio, "Fault tolerance management in cloud computing: A system-level perspective," IEEE Systems Journal, 7(2), pp.288-297, 2013.

[8] S. Misra, A. Gupta, P.V. Krishna, H. Agarwal, and M.S. Obaidat, "An adaptive learning approach for fault-tolerant routing in Internet of Things," In Wireless Communications and Networking Conference (WCNC), pp. 815-819, IEEE, 2012.

[9] W. Zhao, P.M. Melliar-Smith, and L.E. Moser, "Fault tolerance middleware for cloud computing," In 2010 IEEE 3rd International Conference on Cloud Computing (CLOUD), pp. 67-74, IEEE, 2010.

[10] P.H. Su, C.S. Shih, J.YL. Hsu, K.J. Lin, and Y.C. Wang, "Decentralized fault tolerance mechanism for intelligent iot/m2m middleware," In IEEE World Forum on Internet of Things (WF-IoT), pp.45-50, IEEE, 2014.

[11] M.A. Razzaque, M. Milojevic-Jevric, A. Palade, and S. Clarke, "Middleware for internet of things: a survey," IEEE Internet of Things Journal, 3(1), pp.70-95, 2016.

[12] N. Mohamed, J. Al-Jaroodi, I. Jawhar, H. Noura, and S. Mahmoud, "UAVFog: A UAV-Based Fog Computing for Internet of Things," in proc. 2017 IEEE Int'l Conference on Scalable Computing and Communications (IEEE ScalCom 2017), pp. 1758-1765, 2017.

[13] B. Tang, Z. Chen, G. Hefferman, T. Wei, H. He, and Q. Yang, "A hierarchical distributed fog computing architecture for big data analysis in smart cities," In Proceedings of the ASE BigData & SocialInformatics, ACM, 2015.

[14] M.N. Cheraghlou, A. Khadem-Zadeh, and M. Haghparast, "A survey of fault tolerance architecture in cloud computing," Journal of Network and Computer Applications, 61, pp.81-92, 2016.

[15] J. Al-Jaroodi and N. Mohamed, "Service-Oriented Middleware: A Survey," in The Journal of Network and Computer Applications, Elsevier, Vol. 35, No. 1, pp. 211-220, Jan. 2012.

[16] N. Mohamed and J. Al-Jaroodi, "A survey on service-oriented middleware for wireless sensor networks," Service Oriented Computing and Applications, 5(2), pp.71-85, 2011.