Remittances Review
June 2024,
Volume: 9, No: 3, pp.1527-1537
ISSN: 2059-6588(Print) | ISSN 2059-6596(Online)

Received: 19 May 2024, Accepted: 25 July 2024
DOI: https://doi.org/10.33282/rr.vx9i2.66

Handling Fault Tolerance in Edge Computing Using Reactive and Proactive Methods

Muhammad Mudassar [*] [1], Uzair Ishtaiq[1], Muhammad Zaheer Akhtar[1]

[1]Computer Science Department, COMSATS University Islamabad Vehari Campus, Vehari, Pakistan
[*]Corresponding author muhammad.mudassar@cuivehari.edu.pk

Abstract

Edge computing provide an effective way to handle the latency oriented Internet of Things (IoT) applications by leveraging the resources of the processing nodes inside the edge network. The device heterogeneity and highly dynamic environment of edge network causes the failures to happen frequently which can result to a service failure or system unavailability. The system availability can be ensured by providing efficient fault tolerance mechanism to IoT applications in general and special for real-time applications as failures can degrade the performance or stop the processing resulting in problems for the user or a real-time system. This article highlights the different fault tolerance mechanisms including reactive and proactive mechanisms for different processes executing on the edge nodes inside the edge network. And for edge computing we have covered both single edge node execution and distributed application execution. In addition, future research directions are discussed for the researchers working on edge computing.

Index Terms—*Distributed computing, Edge Computing, Fault tolerance, Internet of Things, Reliability.*

Remittances Review
June 2024,
Volume: 9, No: 3, pp.1527-1537
ISSN: 2059-6588(Print) | ISSN 2059-6596(Online)

I. INTRODUCTION

The evolution of internet and development of powerful smart devices helped to build a bigger network known as Internet of Things (IoT). The IoT is providing an ease to individuals by presenting lot of applications including smart cities, home automation, smart security surveillance, and smart health care. A user can enjoy these IoT applications at any time of the day and from any place. The works such as micro datacenter [1] [2], mobile edge computing [3], cloudlet [4], and fog computing [5] has are presented by research community to process IoT data generated by the end devices closer to user space. Because the cloud computing is not always effective for data processing, especially, when data is produced at the edge of the network. The edge computing is identical to fog computing [6], but edge computing is mostly things oriented, while the fog is infrastructure oriented. In an edge computing environment the things can act as data producers and consumers (sensors/actuators), and at the edge of network (edge nodes or smart devices) can perform computations for the application tasks as shown in Fig. 1.

Smart devices referred to as nodes in edge computing can offer storage and cache of data, processing of data (smart devices), offload their computation on requirement, and distribute incoming requests to the cloud. This allows real-time processing to happen, and the time critical applications can be deployed inside the edge network to perform decision making without any delay. Additionally, instead of transmitting entire data to the cloud, the edge device can filter the incoming data to obtain some initial useful information, thus extenuating the stress in backhaul links, resulting in efficient bandwidth utilization.

Usually, smart devices (edge nodes) are co-located in an area during a period, intimating these devices can potentially collaborate to process a resource demanding, real-time, delay sensitive, big task. The distributed processing of a compute-intensive task (e.g., face recognition, object detection, geographical navigation, object tracking and artificial intelligence) on the different devices in the edge network closer to end devices influence in lower latency. This distributed processing on edge devices can enhance the computing capabilities of edge network and help to face challenges like efficient resource utilization, limited bandwidth, reduce overall delay, scalability and fault tolerance. The resource leveraging at the edge of an IoT network can enhance the robustness and dynamicity for the edge computing environment.

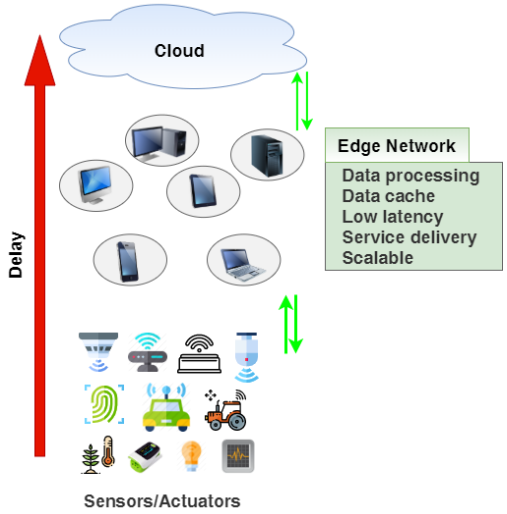

Fig. 1: Edge computing environment in IoT

Fault tolerance is important for many IoT applications in general and special for real-time applications because failures can result in disrupt of a critical activity, which can be dangerous for a user or a real-time system. The devices in the edge network of IoT systems are deployed in an individual with different capabilities and requirements. The lack of redundancy plan for smart devices and gateways during their deployment phase makes fault tolerance very important issue for the edge computing environment. The need for a reliable fault tolerance system reduces the risks to a minimal. For the edge computing fault tolerance should be dynamic, which retains the connected systems together, endure reliability, and availability for the overall system executing an IoT application.

The rapid advancements in edge computing have significantly benefited IoT applications by providing users with quicker results while maintaining quality of service (QoS). However, ensuring service reliability and availability remains a challenge for both service providers and end users. The distributed, decentralized, and dynamic nature of edge computing, coupled with the heterogeneity of edge devices, can lead to various errors and failures in the edge network, resulting in performance degradation. The key types of faults in edge computing are summarized below:

- **Device faults:** These include failures such as node crashes, limited service availability due to low power, and hardware issues like CPU, memory, storage, or sensor/actuator malfunctions. Other problems include communication port failures, devices going out of range, and lack of hardware support.
- **Network faults:** Examples include link failures, network partitioning, congestion, communication errors, timeouts, packet loss, and out-of-range errors due to device mobility.
- **Services faults:** These occur when specific services are unavailable on edge nodes due to resource shortages, software issues, or deadlocks caused by physical or external world dependencies.
- **Other faults:** Issues such as unsupported migration/offloading, environmental hazards, and

sensor/actuator mismatches fall into this category.

Failures in any system can lead to service disruptions or complete shutdowns. In edge computing, partial failures are more common, such as a single device or service failing, which makes centralized fault management impractical. A decentralized fault tolerance mechanism can mitigate these challenges, improving system efficiency and ensuring reliability and availability while minimizing service disruptions [7].

This paper makes following contributions:

- It will provide an overview of IoT applications and their key properties.
- It provides a comprehensive study of distributed application execution in edge computing environment.
- Provide a comprehensive study of fault tolerance techniques and their importance.
- An effective and to the point study of different methods of fault tolerance used in the edge/fog. A through discussion of each technique, faults and errors covered, and their likely causes on edge side processing.
- Along the understanding provided by the paper and discussions about challenges and solutions, a future research direction is provided.

The objective is to provide insight to exiting methodologies used for fault-tolerant solutions, and still what challenges needed to be tackled. To best of our knowledge studying fault tolerance for edge nodes based on their resources and propertied is the recent one in the recent years.

II. Background and Basic Concepts

In this section we recall the basics and features of IoT applications. Latterly, we will review the edge computing environment. Finally, we will provide an overview of current efforts to process tasks distributed in the edge computing environment.

A. IoT Application Features

The Internet of Things (IoT) refers to the network of smart phones, smart machines, medical implants, and other physical objects having sensing/actuation, computing, storage and communication capabilities, this has empowered people and applications to share real-time information to each other and to the physical world [8]. IoT projected to bring together billions of smart devices and smart things, by linking them like the internet done so far with information and computers. This organization of smart objects will enable new forms of interaction among things and people. With the growth of IoT an increasing number of IoT applications is observed, and these applications covering a variety of scenarios, including smart cities, smart transportation, smart grid, security surveillance and smart health care [9 –11].

IoT branded by huge set of distributed objects termed as ‘‘things’’ with limited storage and processing capacity aiming to provide efficiency, reliability, and privacy [12]. However, typical IoT applications demand mobility support, location-awareness, high availability, geo-distribution, and low latency [13].

Most of the IoT applications are composed of independent and distinct modules, which can easily be deployed on separate compute nodes, where the computations can be done on the nodes in the edge or at cloud [14]. In a usual IoT application, part of the business logic is offloaded to the edge of network which results reduce communication overhead, latency and increase application robustness [15]. Such IoT applications are frequently processed distributed on low cost nodes connected to sensors, and some compute nodes at cloud.

Another property of IoT application and devices is that they are generating huge data. This big data can be very fruitful if processed and analyzed correctly on time. The typical big data processing techniques prefer to use cloud computing resources as these are huge set of resources, but for an IoT environment using cloud resources from end user devices results in delay, high bandwidth consumption. Hence, this scenario is not suitable solution for applications requiring real-time analysis. In the literature [5] [16] it is widely acknowledged that cloud computing is not viable for most of the IoT applications requiring real-time processing, for such applications edge computing could be used as an alternative, this will help real-time processing and reduce networks usage.

Based on the characteristics of IoT applications, several key features are identified:

- **Real-time interaction:** Many IoT applications require real-time interaction, such as in healthcare, traffic monitoring, or security systems.
- **Low latency:** Timely communication and processing are crucial for improving QoS and meeting essential requirements.
- **Geographical distribution:** IoT deployments often span large areas to serve both stationary and mobile users.
- **Support for mobility:** Mobility is critical for applications that require communication with mobile devices or serve users on the move.
- **Location awareness:** Identifying object positions is essential for applications like security and surveillance systems.
- **Fault-tolerant:** Efficient fault-tolerant mechanisms are needed to address device failures or unavailability.
- **High Task Demand:** Continuous data streams from end devices necessitate high-end processing for some IoT tasks.
- **High communication required:** Applications such as video streaming or social networking often require significant data exchange.
- **Device Heterogeneity:** IoT applications must operate across a wide variety of devices with differing hardware and software configurations.

Remittances Review
June 2024,
Volume: 9, No: 3, pp.1527-1537
ISSN: 2059-6588(Print) | ISSN 2059-6596(Online)

B. Edge Computing Environment

In edge computing, the processing is performed at the edge of network on different devices such as smart phones, smart sensor nodes with processing attached to them, wearables devices, and on-board units. The data analytics and knowledge generation can be accomplished on these edge nodes removing the need for a centralized system [17]. Edge computing excelled the technology world due to its tremendous performing competences in terms of providing real-time analytics, cost effective processing, high scalability, reduced delay and latency while offering an improved quality of service (QoS) [11]. Edge computing technology that suggests to empower the network edge will reform numerous fields like education, healthcare, planning and management of city area services, e-commerce and social networks.

Edge can be defined as any computing resource or a network device capable to process data from source to cloud. For example, a smart mobile phone can act as edge between user and cloud to process user applications, a gateway device (network device) in a smart home is an edge node between home appliances and cloud. The principle of edge computing is that computing should performed be in closeness of data sources and at the edge of the network. This will help to address concerns of the latency requirement of IoT applications, fulfill battery power limitation, bandwidth saving. The data privacy and security can also be achieved.

The edge computing environment allows computing at the vicinity of data sources at edge of the network. This results in numerous benefits when compared to the cloud based computing. Some early results from the research findings reveal the potential benefits of edge computing. Researchers constructed a framework to run face recognition application in [6], and declared response time decreased from 900 to 169 ms by pushing computations to edge of the network. Ha et al. [18] used cloudlet technology to offload the executing tasks for wearable cognitive support; the overall improvement in response time between 80 and 200 ms is being claimed. Furthermore, the energy consumption improvement is obtained by 30% to 40% by cloudlet offloading. Clonecloud in [19] combines several factors like partitioning, job migration, and on-demand instantiation of partitioned tasks between mobile node and cloud, and they showed that their proposed methodology results in reduction of 20% of running time and energy.

In an edge computing environment, a design by Vallati et al. [20] achieves a remarkable reduction in latency and promises the security of locality information. Smart city solution designed by [21] uses edge computing for the smart city application that can effectively identify certain dangerous events related to a city environment, such as terrorist threats, natural disasters, fabricated disasters, etc. Research work in [22] and [23], proposed to designed an architecture based on edge computing to address several network-related matters like an efficient offloading scheme to reduce computation complexities in vehicular technology.

Table 1: Comparison of computing characteristics for IoT applications

Characteristic	Edge computing	Mobile cloud	Cloud
Application latency	Low (Milli Sec)	Medium (S-M)	High (M-H)
Bandwidth	Very low	Medium	High
Response time	Low	High	High
Resources	Limited processing and storage	Medium computation and storage	Ample computation and storage
Scalability	High	Medium	Low
Energy depletion	Low	Medium	High
Quality of Service (QoS)	High	Medium	Medium
Deployment	Dist. & Decent.	Centralized	Centralized

We have presented a comprehensive analysis of different contributions of edge computing across different fields by plentiful researchers certifying edge computing to be a truly reliable and available computing system, in an efficient way and using decentralized manner. We have provided specifics advantages of edge computing over other similar domains used to process IoT applications; a comparison is given in Table 1. We are providing a comparative analysis of the various characteristics related to computing when performed in edge computing, mobile cloud computing and cloud computing. It shows that edge computing is better than other similar approaches used for processing IoT applications.

C. Distributed Processing in Edge Computing Environment

The execution environment of the edge computing tries to execute application tasks locally near to the user space before processing at cloud, resulting in decreased network overhead consequences, application delay, data security and privacy matters. Comparing the plentiful of resources at the cloud processing nodes in the edge network are low power devices and heterogeneity assassinated with them, along with device mobility. Executing resource intensive IoT applications on individual edge nodes can hamper the quality of service (QoS) and user satisfaction. However, distributed execution on the smart devices available in the proximity can successfully execute a resource demanding, real-time, latency-oriented task by dividing its workload among available devices. This will allow the applications to take advantage from the edge computing environment as well.

Research exist suggesting to combine edge computing resources including Cloudlet [24], femtoclouds [25] and Fog computing [6]. The clustering concept in Femtoclouds is based on using dedicated controllers while following a centralized mode [26], which can bottleneck the centralized computing entity resulting in services degradation. Research work in [27] proposed clustering methodology for mobile

edge computing (MEC) by using a graph-based approach, their methodology tries to handle most of the communications at the edge to limit the overall network traffic. MEC clusters are formed by segregating geographical areas restricting the communication inside a cluster resulting in cutting load to the cloud.

To leverage computing resources of edge nodes in an edge computing environment, mobile edge clouds [28], are proposed to coordinate numerous edge devices for resource intensive applications that are difficult to execute successfully on single edge device. Firework [31] leverages mobile devices and the cloud to process big tasks, it also combines different edge nodes to accomplished big data processing tasks cooperatively. Authors in [32] proposed a distributed wireless surveillance system, their approach prioritizes video frames relevant to query performed by user, and achieves maximum objects required by the query and at same time decreasing the cost incurred for wireless bandwidth. Researchers in [33] proposed an open-source framework named as OpenFace, their framework can be used for face recognition at a real-time as well as it can perform tracking operation by using edge computing.

III. FAULT TOLERANCE TECHNIQUES

Fault tolerance is extremely important for edge computing to provide reliable services to the IoT applications executing in edge computing environment. Different fault tolerance methodologies are used for identifying different faults and handling these faults in the system that may happen due to device faults, network faults, services faults or any other faults. Handling a fault efficiently results in robustness of the system. Different fault tolerance approaches are used in the literature can be categorized as reactive or proactive approaches as shown in Fig.2.

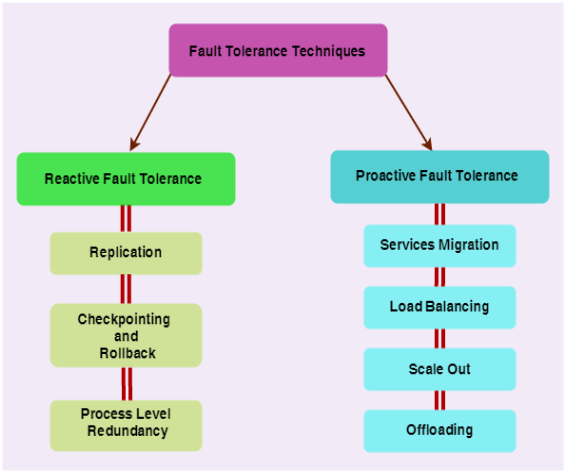

Fig. 2: Fault tolerance techniques for the edge computing

A. Reactive Fault Tolerance

The reactive techniques are primarily used to reduce the effect of failure after it happens in the system. Some reactive fault tolerance techniques are explored in following after a thorough review of literature related to distributed systems, cloud and IoT systems. These techniques are cost effective and can be fruitful for the edge computing environment.

Replication: In replications based fault-tolerant syste critical parts (e.g. process, data, and communication paths) of the system are replicated using redundancy techniques, and when the working system fails, the backup takes over to keep the system working [34]. A task can continue its processing in presence of failures or errors until there exist a replica in the system.

Checkpoint and Rollback: In checkpointing and roll back technique, the current state of the system is periodically stored to a backup node known as checkpoint [35]. This checkpoint information at later used for rollback the computations after failure of the node. The checkpoint file includes environment variables, process state, values of registers and other useful information to restore the system to a stable state [36].

Process Level Redundancy: The process level redundancy is applied where deploying hardware related fault tolerance techniques become more expensive. This method compares processes to ensure accurate execution and it generates a set of redundant processes for each of the application process [37].

Table 2 presents a comparison of different features for each of the mentioned technique.

Table 2 : Comparison of different techniques used for fault-tolerant system

Comparison property	Replication	Checkpoint and rollback	Process level redundancy
Functioning	Create a separate replica of important entities	Store working state to stable place for recovery	Create a set of redundant processes
Performance	Depends on number of replicas, less replicas high performance	Checkpointing interval and file size, efficient for low frequency and small size	Decreases as frequency of fault and repair increases
Fault handling	More number of replicas more faults are tolerable	Depends on efficient chickening scheduling mechanism	Depends on scaling number of redundant processes
System consistency	More replicas can cause problem to system consistency	More consistent if global checkpointing is used	Process redundancy to easily schedule process on hardware
Cost effective	Increase cost with number of replicas increased	Cost effective solution	Enhancing process level redundancy need more resources

B. Proactive Fault Tolerance

This methodology tries to avoid from faults and errors to occur. Cause of faults are determined at prior and a substitute component is placed to keep the system smooth execution. Following an overview is provided for different proactive approaches.

Services Migration: A service or node is observed continuously and pre-copy service/VM migration is performed based on some predefined criteria to avoid a failure or an error [38].

Load Balancing: This methodology is used to balance the load of machines between different available machines when it exceeds a certain perimeter [39].

Scale Out: Identifying the bottleneck node and then scaling the application by dividing the workload and increasing the processing nodes to ensure the application smooth working in case of node approaches to a bottleneck [40].

Offloading: a terminology frequently used in IoT and edge computing environments. Offloading a task to a remote server assure to bridge the gap between limited processing capability of edge node and high computation requirements of a resource intensive IoT applications [15], [70].

IV. FAULT TOLERANCE FOR EDGE COMPUTING ENVIRONMENT

Designing a reliable, efficient and effective fault-tolerant system for the edge computing is a vital issue, primarily due to heterogeneity and large diversity present among edge devices, networks and methodologies used for data processing [42]. The significant point is to keep IoT application operational if any component of the IoT system goes off or faulty. A fault-tolerant system should provide availability and reliability [34]. Availability is concerned with the system that it is ready to deliver its functionality during defined period, while reliability refers to property of the system to function unbrokenly provided a specific time interval.

Here at first we will provide review of fault tolerance methodologies used for individual devices in the edge computing environment. Afterwards, we will discuss in detail about fault tolerance for the distributed edge computing environment.

A. Fault Tolerance for Individual Device in Edge Computing

1) Reactive Approaches

The process of replication adds redundancy in the system, and when smart devices are used as edge nodes the redundancy can be increased by utilizing devices in the vicinity. Fault tolerant health monitoring system is discussed in [43] it uses the concept of redundant devices and implements an enhanced gateway for fault tolerance. According to this research work device preserves a consistent view by duplicating the services a failure can be recovered within a short interval without any external interference.

The research work [44] proposed to use an agent-based architecture for IoT by following a hierarchical architecture, to ensure reliability and fault tolerance. They have used the mobile agent to monitor the resources and network. The data replication at edge of the network helps to provide the reliability, a redirection is performed when a failure happens. The possible level for redirection is among cloud, fog, mist or dew. To handle unexpected faults, the agent will get the priority index for all applications executing on the failed edge node and checks for available nodes at the same level once find the application migration will be performed and connection rerouting is done. In [45] authors discussed the fault tolerance and reliability concerns for the fog computing to support smart city applications. Fault tolerance is achieved using a replica of services in the fog, upon failure of a fog node the services that are processed by this node are replaced by similar service available on another working fog node available in the locality.

Some research proposed to build energy-efficient fault-tolerant approach for specific devices in IoT by checkpointing the program execution data to stable storage on the same device [46], [47]. Main point is to store selected states of a program to reduce the time overhead involved in writing checkpoint data to Non-Volatile Memory (NVM); they have used a well-known algorithm “max-flow min-cut” on the data flow model of the program. Their focus is on fault recovery for a single device.

Edge devices face resource contention making it tough to handle IoT applications especially during the disasters, for

situations like this fault tolerance is must required in order to cope with the situation,\a distributed task offloading scheme is provided by [41] to ensure fault tolerance for mobile edge computing networks. Their approach reduces task execution time and system energy by using multi-agent proximal optimization algorithm. The system is suitable for failure scenarios occurred during services to meet the task requirements.

A checkpointing mechanism known as Distributed Multi-Threaded Check Pointing (DMTCP) offers a methodology for transparent checkpointing, this technique performs checkpoint/restart without any change to original code of the application or operating system, it can be adapted for IoT applications [48]. The scene here represents a partial view of the global state is check pointed, and after failure occurs a new scene in the form of a process resumed from a checkpoint file.

2) Proactive Approaches

Research work in [49] proposes an intelligent framework that is based on semantic reasoning. They have proposed to use checkpointing mechanism to handle more flexibly an IoT system, where gateways are installed to collect data and connect to cloud for user application. They handled dynamicity, which can occur due to new services or mobility by means of a checkpointing methodology, hence, migration of the software can be performed from one gateway to another gateway. This will help in cases such as transportations, logistics, or applications where devices need to change their physical position to some other place or city. Additionally, this will also help when the gateway device has limited resources in terms of memory, processing, battery power, and other resources.

Authors in [50] have proposed a microservices based fault tolerance framework to provide real-time and predictive fault tolerance support to IoT systems. Their framework uses two microservices to provide reactive and proactive fault tolerance. One microservice “Real-Time FT” uses complex event processing (CEP) and analyzes incoming streams of data for quick error recovery. The second one named “Predictive FT”, which uses machine learning techniques to allow the system to learn and identify when a fault begins or likely to start due to similar prior learned experiences and mitigate the future faults. This will ensure the proactive approach.

Authors in [68] provided a modified (m, n)-fault tolerance strategy named as M-MNFT, their strategy is different from others is that their method take in to account failure of the edge server, moreover their approach selects some other edge base stations available redundantly to provide task migration reliability, this is based on the reduced relative distance fact between the source and destination edge server wile sending the request this will mitigate impact of the failure of the edge base station on QoS during the task transfering.

B. Fault Tolerance for Distributed Edge Computing

The resource intensive big tasks can also enjoy benefits of edge computing in terms of locality processing with reduced latency and lesser network traffic by distributing the processing on the available nodes in the edge network. However, providing failure handling will become crucial for such a scenario. Different approaches have potential to provide fault tolerance for the distributed edge computing including checkpointing, replication, offloading and other methodologies discussed in Section 3, as shown in Figure 3. For the smart devices present in the edge network with limited resources these techniques with modification can result in an efficient fault-tolerant system for distributed edge computing environment. In following, we will discuss different methodologies and systems used by the research community to provide fault tolerance for the distributed edge computing.

Cluster formation among devices in IoT to execute a task efficiently for a given application, the clustering helps in device collaboration and helps in prolonging overall network lifetime [32]. One common methodology used to solve the fault-tolerant problem for clustering is to use dominate set clustering [51 – 53], where it is tried to find dominate set inside a specific network thus every node of the network is within k hops to the nodes present in the dominate set. LEACH [54] have proposed a distributed algorithm to elect a cluster head (CH) based on energy level of each node in comparison to neighbors, the cluster head could be switched as energy changes or a head node fails to help with load balance and provide fault tolerance.

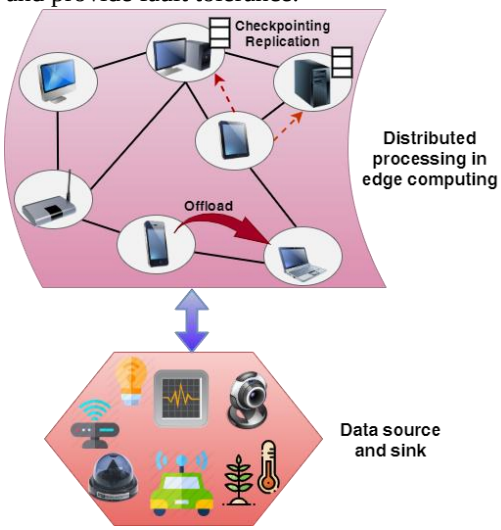

Fig.3: Fault tolerance methodologies for distributed edge computing environment

All of these involve the extra overhead of cluster formation and management, which is an extra burden on the cluster head. The benefit of a centralized methodology is that it can get lot of information that is precise and a system that is more powerful. Among the centralized techniques, research work in [55] tries to attain fault tolerance for clusters by allocating nodes to some existing gateway devices by following a heuristic based algorithm.

In edge computing main concern is with distributed and decentralized patterns with heterogeneous devices. The framework in [56] optimizes the fault tolerance for service based IoT applications in virtualized wireless sensor networks, with an emphasis on heterogeneity present in

networks. The benefits of distributed methodologies mainly reside on that it could better adjust to the mobility of smart devices used as edge nodes, and better scalability [51]. Data replication is a promising technique to preserve valuable sensed data in a distributed network. DRAW [57] offers a fully distributed data replication technique by replicating data hop-by-hop for IoT systems. This guarantees maximum data availability even under high node failure rate, but in a limited node environment, the number of distinct replicas of data that can be stored in the network decreases. It also suffers the overhead of extra messages transferred between nodes to create replicas on these nodes.

To ensure fault tolerance and automated recovery can be performed by using existing well-known technologies can also be used like Containers, Kubernetes, and Apache Kafka. CEFIoT presents and architecture for a fault-tolerant system to execute IoT applications in small edge clusters and cloud [58]. They used Apache Kafka for data replication solution and Kubernetes for fault-tolerant management to provide on-the-fly dynamic reconfiguration of the processing array to handle failures.

Crystal provides an easy abstraction for fog application development to build a sustainable distributed fog computing application [59]. Their implementation of crystal using MapReduce framework achieved fault tolerance for distributed processing over heterogeneous, unreliable, fog nodes. They also showed to reduce overall latency by processing data close to the source.

Designing an efficient fault-tolerant system also involves investigating the reliability parameters of computing devices in the distributed system. The reliability and availability modeling is a very important requirement to ensure robust design and operations. A fault-tolerant and energy efficient framework is proposed for remote storage of data and processing of tasks under a dynamic network topology of a mobile cloud [60]. They used a k-out-of-n reliability mechanism for distributed computing of mobile cloud to partition data and object to store on different nodes, until k or more out of n nodes are working a failure can be recovered. Fault tolerance using the redundancy technique is used to meet high reliability and availability requirement of SAN [61], they have evaluated the reliability of mesh SAN by using a binary decision diagrams.

Fault tolerance can be achieved for IoT through virtual service composition, where using single service executing on single devices with backup devices for each other [62]. There exists natural redundancy of services across different devices available at office, home, and different scenarios; this should be exploited to provide fault tolerance [63]. Research work in [64] proposes a fault-tolerant platform for smart home applications, mainly concentrating on link failure. It provides fault-tolerant event transfer of sensor and actuation commands in the presence of link failure and network partitions. An IoT based architecture for health care environment with fault tolerance is presented in [43], their approach considers network fault tolerance by using backup routing between nodes and advanced service mechanisms to ensure connectivity in case of error or a failure to a connection.

A decentralized distributed fault tolerance methodology is presented in [7]. This research work at first presented a methodology for edge node group formation to execute a resource intensive task in parallel and distributed on the nodes in the group. This will help to achieve the application latency requirements. They have calculated the edge node reliability parameters based on device local properties, and used these reliability calculations to provide an efficient fault tolerance methodology. The fault tolerance methodology replicates process and data on the set of neighbor nodes to ensure the availability and reliability parameters calculated previously.

Authors in [65] provided a distributed fault tolerant system to handle dynamic IoT environment. A strip data structure is designed to manage replicated services; this provides a redundancy abstraction for service peers. Each device maintains consistent view of replicated services on strip. The heartbeat protocol and manipulation of strip in distributed manner allows to recovery from failure. The research work in [66] present the implementation of a platform to ensure mobility and reliability to computational tasks executing in mobile cloud platforms. Replication is used for fault tolerance and by placing redundant processing on different nodes is achieved in a distributed system. They have also handled easy migration of tasks executing in the access network and provided the necessary functions.

The table 3 provides a detailed comparison of different methodologies and frameworks used for fault tolerance in the IoT edge computing environment. The different features are compared with focus on network approach followed as centralized or decentralized, it is clear that most of the distributed fault tolerance research works have followed a centralized methodology and a fewer has the decentralized one. The other feature compared is about which technique is followed for making the backup in the system. Replication and checkpointing techniques are massively used by the research community to design a fault tolerance for the edge computing environment. Most of the literature have not considered the device heterogeneity that is a key property of devices present in the edge network. A very few works have calculated the reliability parameters. Most of the techniques are concerned to provide fault tolerance to the process and data of an IoT application.

Table 3: A comparison of fault tolerance approaches for IoT systems

Literature	Distributed	Approach	Consider heterogeneity	Supports	Reliability Measures	Technique	Description
Energy [46]	No	I	No	Process and data	No	CP	Support single device failure in IOT
Mobility	No	D	No	Process	No	CP	Support mobility of gateway device

[49]				and data			
IoT/M2M [65]	Yes	D	No	Services	No	RP	Automatic recovery of failed service without user interventions
Mobile cloud [60]	Yes	C	No	Process and data	Yes	RP	Support dynamic network topology
Mesh SAN [61]	Yes	C	No	Communication	Yes	RD	Ensure link reliability of SAN
Hybrid [67]	Yes	C	No	Process	No	Hybrid	Use reactive and proactive policies for FT
Smart city [45]	Yes	D	Yes	Service	No	RP	Consider failure of fog node to provide FT to smart city applications
FaaS [29]	Yes	C	Yes	Microservice	No	CP	Use function as a service (FaaS) and checkpointing for long running functions
DRAW [57]	Yes	D	No	Data	No	RP	Increase data replication to keep IoT application working
Crystal [59]	Yes	D	Yes	Process & communication	No	RD	Framework for designers to follow MapReduce functionality on crystals
Resilience [30]	Yes	CS	Yes	Process	No	CP	Restore process and interaction with physical world of devices in failure
Legend: C=centralized, D= Decentralized, I=Individual, CS= Client-Server, CP=checkpointing, RP= Replication, RD=Redundancy							

V. FUTURE DIRECTIONS

Edge computing, being in its infancy stage, has already attracted the research community and IoT industry, and it is predicted to be the major driving force for the latency-oriented processing. The fault tolerance in edge computing environment still has a number of challenges due to its architecture.

A. Offloading

An efficient offloading model is required for processes executing on the edge nodes. This will help to handle individual edge node faults due to low power and processing resource limitations of an edge node. The offloading approach requires optimization in terms of performance in the real-time scenario.

B. Optimized Resource Allocation with Fault Tolerance

The fault tolerance system has to consider the resource allocation optimizations to achieve an effective reliable fault-tolerant system. The device heterogeneity has to be handled while the resources allocation is performed, this will result in task allocation optimization and based on the available resources of a node a task can finish well in time based on the available resources of the device. This will result in fewer task failures.

C. Fault Tolerance and Decentralized Methodology

The nature of edge computing is distributed and decentralized, hence, if the fault tolerance methodology follows the fault handling using a decentralized mode based on individual device properties it can result in an efficient system.

D. Reliability Measurements

The fault tolerance system should be based on proper reliability measures. This can be achieved by measuring the individual node failure probabilities, and other related parameters like device failure rate, and meat time between failures (MTBF).

E. Energy Efficiency

An efficient fault tolerance system that can significantly achieve energy efficiency at same time. Extending cloud to the edge of the network will involve deploying edge devices closer to the user, the more devices deployed more energy will be required. A fault tolerance system considering the energy of devices is required. For this opting an adaptive checkpointing mechanism can be a viable solution.

F. Fault Tolerance for Limited Devices

Due to mobility property and resource limitations, devices in the edge network might be inadequate to provide backup for every entity of the IoT system. An efficient fault tolerance system is required that should select the backup techniques based on the available resources in the edge network, and should be dynamic with respect to available resources present in the edge network.

CONCLUSION

The edge computing could help to achieve latency requirements of the applications by executing them closer to the user at edge of network. Failures are inevitable in edge nodes due to their mobility property and resources limitations. To guarantee successful execution of critical IoT application running in an edge network, fault tolerance becomes a vital issue. An efficient fault tolerance technique

helps to achieve reliability and availability in the system along with handling failure recovery. This survey paper has provided a detailed overview of edge computing environment with special focus on the fault tolerance. Various fault tolerance techniques are outlined, which are being used in the edge computing environment to design a fault tolerance system. Further, fault tolerance for the distributed edge computing environment is reviewed. By comparison of current methodologies, we have put forth some future directions for the research initiatives.

REFERENCES

[1] A. Greenberg, J. Hamilton, D. Maltz, and P. Patel, "The cost of a cloud: Research problems in data center networks," Computer Communications Review, vol. 39, no. 1, January 2009. [Online]. Available: https://www.microsoft.com/en-us/research/publication/the-cost-ofa-cloud-research-problems-in-data-center-networks/

[2] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: Making smartphones last longer with code offload," in ACM MobiSys 2010. Association for Computing Machinery, Inc., June 2010. [Online]. Available: https://www.microsoft.com/en-us/research/publication/mauimaking-smartphones-last-longer-with-code-offload/

[3] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "Mobile edge computing: Survey and research outlook," arXiv preprint arXiv:1701.01090, 2017.

[4] [4] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," IEEE Pervasive Computing, vol. 8, no. 4, pp. 14–23, Oct 2009.

[5] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in Proceedings of the first edition of the MCC workshop on Mobile cloud computing. ACM, 2012, pp. 13–16.

[6] S. Yi, Z. Hao, Z. Qin, and Q. Li, "Fog computing: Platform and applications," in 2015 Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb). IEEE, 2015, pp. 73–78.

[7] M. Mudassar, Y. Zhai, L. Liao, and J. Shen, "A decentralized latency-aware task allocation and group formation approach with fault tolerance for iot applications," IEEE Access, vol. 8, pp. 49 212–49 223, 2020.

[8] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (iot): A vision, architectural elements, and future directions," Future Generation Computer Systems, vol. 29, no. 7, pp. 1645 – 1660, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X13000241

[9] F. Dalipi and S. Y. Yayilgan, "Security and privacy considerations for iot application on smart grids: Survey and research challenges," in 2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW). IEEE, 2016, pp. 63–68.

[10] S. Latre, P. Leroux, T. Coenen, B. Braem, P. Ballon, and P. Demeester, "City of things: An integrated and multi-technology testbed for iot smart city experiments," in 2016 IEEE International Smart Cities Conference (ISC2). IEEE, 2016, pp. 1–8.

[11] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," IEEE Internet of Things Journal, vol. 3, no. 5, pp. 637–646, Oct 2016.

[12] A. Botta, W. [de Donato], V. Persico, and A. Pescape, "Integration of ´ cloud computing and internet of things: A survey," Future Generation Computer Systems, vol. 56, pp. 684 – 700, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X15003015

[13] S. Yi, Z. Qin, and Q. Li, "Security and privacy issues of fog computing: A survey," in Wireless Algorithms, Systems, and Applications, K. Xu and H. Zhu, Eds. Cham: Springer International Publishing, 2015, pp. 685–695.

[14] R. Jain and S. Tata, "Cloud to edge: Distributed deployment of process-aware iot applications," in 2017 IEEE International Conference on Edge Computing (EDGE), June 2017, pp. 182–189.

[15] H. Flores, P. Hui, P. Nurmi, E. Lagerspetz, S. Tarkoma, J. Manner, V. Kostakos, Y. Li, and X. Su, "Evidence-aware mobile computational offloading," IEEE Transactions on Mobile Computing, vol. 17, no. 8, pp. 1834–1850, 2017.

[16] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, Fog Computing: A Platform for Internet of Things and Analytics. Cham: Springer International Publishing, 2014, pp. 169–186.

[17] H. El-Sayed, S. Sankar, M. Prasad, D. Puthal, A. Gupta, M. Mohanty, and C. T. Lin, "Edge of things: The big picture on the integration of edge, iot and the cloud in a distributed computing environment," IEEE Access, vol. 6, pp. 1706–1717, 2018.

[18] K. Ha, Z. Chen, W. Hu, W. Richter, P. Pillai, and M. Satyanarayanan, "Towards wearable cognitive assistance," in Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services, ser. MobiSys '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 68–81. [Online]. Available: https://doi.org/10.1145/2594368.2594383

[19] H. Gedawy, K. Habak, K. Harras, and M. Hamdi, "An energy-aware iot femtocloud system," in 2018 IEEE International Conference on Edge Computing (EDGE), July 2018, pp. 58–65.

[20] C. Vallati, A. Virdis, E. Mingozzi, and G. Stea, "Mobile-edge computing come home connecting things in future smart homes using lte device-todevice communications," IEEE Consumer Electronics Magazine, vol. 5, no. 4, pp. 77–83, Oct 2016.

[21] M. Sapienza, E. Guardo, M. Cavallo, G. La Torre, G. Leombruno, and O. Tomarchio, "Solving critical events through mobile edge computing: An approach for smart cities," in 2016 IEEE International Conference on Smart Computing (SMARTCOMP), May 2016, pp. 1–5.

[22] J. Feng, Z. Liu, C. Wu, and Y. Ji, "Ave: Autonomous vehicular edge computing framework with aco-based scheduling," IEEE Transactions on Vehicular Technology, vol. 66, no. 12, pp. 10 660–10 675, Dec 2017.

[23] K. Zhang, Y. Mao, S. Leng, Y. He, and Y. ZHANG, "Mobile-edge computing for vehicular networks: A promising network paradigm with predictive off-loading," IEEE Vehicular Technology Magazine, vol. 12, no. 2, pp. 36–44, June 2017.

[24] M. Satyanarayanan, V. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," IEEE Pervasive Computing, 2009.

[25] K. Habak, M. Ammar, K. A. Harras, and E. Zegura, "Femto clouds: Leveraging mobile devices to provide cloud service at the edge," in 2015 IEEE 8th International Conference on Cloud Computing. IEEE, 2015, pp. 9–16.

[26] K. Habak, E. W. Zegura, M. Ammar, and K. A. Harras, "Workload management for dynamic mobile device clusters in edge femtoclouds," in Proceedings of the Second ACM/IEEE Symposium on Edge Computing. ACM, 2017, p. 6.

[27] M. Bouet and V. Conan, "Mobile edge computing resources optimization: A geo-clustering approach," IEEE Transactions on Network and Service Management, vol. 15, no. 2, pp. 787–796, 2018.

[28] N. Fernando, S. W. Loke, and W. Rahayu, “Computing with nearby mobile devices: A work sharing algorithm for mobile edge-clouds,” IEEE Transactions on Cloud Computing, vol. 7, no. 2, pp. 329–343, April 2019.

[29] P. Karhula, J. Janak, and H. Schulzrinne, “Checkpointing and migration of iot edge functions,” in Proceedings of the 2nd International Workshop on Edge Systems, Analytics and Networking, 2019, pp. 60–65.

[30] U. Ozeer, X. Etchevers, L. Letondeur, F.-G. Ottogalli, G. Salaun, and ¨ J.-M. Vincent, “Resilience of Stateful IoT Applications in a Dynamic Fog Environment,” in EAI International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous ’18), New York, United States, Nov. 2018, pp. 1–10. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01927286

[31] Q. Zhang, Q. Zhang, W. Shi, and H. Zhong, “Firework: Data processing and sharing for hybrid cloud-edge analytics,” IEEE Transactions on Parallel and Distributed Systems, vol. 29, no. 9, pp. 2004–2017, Sep. 2018.

[32] J. S. Kumar and M. A. Zaveri, “Clustering approaches for pragmatic two-layer iot architecture,” Wireless Communications and Mobile Computing, vol. 2018, 2018.

[33] J. Wang, B. Amos, A. Das, P. Pillai, N. Sadeh, and M. Satyanarayanan, “A scalable and privacy-aware iot service for live video analytics,” in Proceedings of the 8th ACM on Multimedia Systems Conference, ser. MMSys’17. New York, NY, USA: Association for Computing Machinery, 2017, p. 38–49. [Online]. Available: https://doi.org/10.1145/3083187.3083192

[34] A. Sari, M. Akkaya et al., “Fault tolerance mechanisms in distributed systems,” International Journal of Communications, Network and System Sciences, vol. 8, no. 12, p. 471, 2015.

[35] A. Khunteta and K. Praveen, “An analysis of checkpointing algorithms for distributed mobile systems,” International Journal on Computer Science and Engineering, vol. 2, 07 2010.

[36] E. N. M. Elnozahy, L. Alvisi, Y.-M. Wang, and D. B. Johnson, “A survey of rollback-recovery protocols in message-passing systems,” ACM Comput. Surv., vol. 34, no. 3, p. 375–408, Sep. 2002. [Online]. Available: https://doi.org/10.1145/568522.568525

[37] N. Xiong, Y. Yang, M. Cao, J. He, and L. Shu, “A survey on fault-tolerance in distributed network systems,” vol. 2, 01 2009, pp. 1065– 1070.

[38] V. Medina and J. M. Garc´ıa, “A survey of migration mechanisms of virtual machines,” ACM Comput. Surv., vol. 46, no. 3, Jan. 2014. [Online]. Available: https://doi.org/10.1145/2492705

[39] M. Nazari Cheraghlou, A. Khadem-Zadeh, and M. Haghparast, “A survey of fault tolerance architecture in cloud computing,” J. Netw. Comput. Appl., vol. 61, no. C, p. 81–92, Feb. 2016. [Online]. Available: https://doi.org/10.1016/j.jnca.2015.10.004

[40] M. Mudassar, Y. Zhai, and L. Liao, “Efficient state management for scaling out stateful operators in stream processing systems,” Big Data, vol. 7, no. 3, pp. 192–206, 2019, pMID: 30994383. [Online]. Available: https://doi.org/10.1089/big.2018.0093

[41] H. Zhang et al., "Decentralized and Fault-Tolerant Task Offloading for Enabling Network Edge Intelligence," in IEEE Systems Journal, vol. 18, no. 2, pp. 1459-1470, June 2024, doi: 10.1109/JSYST.2024.3403696.

[42] H. Zhang *et al*., "Decentralized and Fault-Tolerant Task Offloading for Enabling Network Edge Intelligence," in *IEEE Systems Journal*, vol. 18, no. 2, pp. 1459-1470, June 2024, doi: 10.1109/JSYST.2024.3403696.

[43] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, “Internet of things for smart cities,” IEEE Internet of Things Journal, vol. 1, no. 1, pp. 22–32, 2014.

[44] T. N. Gia, A.-M. Rahmani, T. Westerlund, P. Liljeberg, and H. Tenhunen, “Fault tolerant and scalable iot-based architecture for health monitoring,” in 2015 IEEE Sensors Applications Symposium (SAS). IEEE, 2015, pp. 1–6.

[45] J. Grover and R. M. Garimella, “Reliable and fault-tolerant iot-edge architecture,” in 2018 IEEE SENSORS. IEEE, 2018, pp. 1–4.

[46] N. Mohamed, J. Al-Jaroodi, and I. Jawhar, “Towards fault tolerant fog computing for iot-based smart city applications,” in 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC). IEEE, 2019, pp. 0752–0757.

[47] T. Xu and M. Potkonjak, “Energy-efficient fault tolerance approach for internet of things applications,” in Proceedings of the 35th International Conference on Computer-Aided Design, 2016, pp. 1–8.

[48] S. Kannan, A. Gavrilovska, K. Schwan, and D. Milojicic, “Optimizing checkpoints using nvm as virtual memory,” in 2013 IEEE 27th International Symposium on Parallel and Distributed Processing. IEEE, 2013, pp. 29–40.

[49] F. A¨ıssaoui, G. Cooperman, T. Monteil, and S. Tazi, “Smart scene management for iot-based constrained devices using checkpointing,” in 2016 IEEE 15th International Symposium on Network Computing and Applications (NCA), Oct 2016, pp. 170–174.

[50] F. A¨ıssaoui, G. Cooperman, T. Monteil, and S. Tazi, “Intelligent checkpointing strategies for iot system management,” in 2017 IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud). IEEE, 2017, pp. 305–312.

[51] A. Power and G. Kotonya, “A microservices architecture for reactive and proactive fault tolerance in iot systems,” in 2018 IEEE 19th International Symposium on” A World of Wireless, Mobile and Multimedia Networks”(WoWMoM). IEEE, 2018, pp. 588–599.

[52] F. Kuhn, T. Moscibroda, and R. Wattenhofer, “Fault-tolerant clustering in ad hoc and sensor networks,” vol. 2006, 02 2006, pp. 68 – 68.

[53] J. Wang, Y. Yonamine, E. Kodama, and T. Takata, “A distributed approach to constructing k-hop connected dominating set in ad hoc networks,” 12 2013, pp. 357–364.

[54] J. Wu and H. Li, “On calculating connected dominating set for efficient routing in ad hoc wireless networks,” in Proceedings of the 3rd international workshop on Discrete algorithms and methods for mobile computing and communications, 1999, pp. 7–14.

[55] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, “Energyefficient communication protocol for wireless microsensor networks,” in Proceedings of the 33rd Annual Hawaii International Conference on System Sciences, Jan 2000, pp. 10 pp. vol.2–.

[56] G. Gupta and M. Younis, “Fault-tolerant clustering of wireless sensor networks,” in 2003 IEEE Wireless Communications and Networking, 2003. WCNC 2003., vol. 3, March 2003, pp. 1579–1584 vol.3.

[57] O. Kaiwartya, A. H. Abdullah, Y. Cao, J. Lloret, S. Kumar, R. R. Shah, M. Prasad, and S. Prakash, “Virtualization in wireless sensor networks: Fault tolerant embedding for internet of things,” IEEE Internet of Things Journal, vol. 5, no. 2, pp. 571–580, 2017.

[58] W. B. Qaim and O. Ozkasap, “Draw: Data replication for enhanced data availability in iot-based sensor systems,” in 2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and

Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech). IEEE, 2018, pp. 770–775.
[59] A. Javed, K. Heljanko, A. Buda, and K. Framling, “Cefiot: A fault- ¨ tolerant iot architecture for edge and cloud,” in 2018 IEEE 4th World Forum on Internet of Things (WF-IoT). IEEE, 2018, pp. 813–818.
[60] T. Jeong, J. Chung, J. W.-K. Hong, and S. Ha, “Towards a distributed computing framework for fog,” in 2017 IEEE Fog World Congress (FWC). IEEE, 2017, pp. 1–6.
[61] C.-A. Chen, M. Won, R. Stoleru, and G. G. Xie, “Energy-efficient faulttolerant data storage and processing in mobile cloud,” IEEE Transactions on Cloud Computing, vol. 3, no. 1, pp. 28–41, 2014.
[62] [61] L. Xing, M. Tannous, V. M. Vokkarane, H. Wang, and J. Guo, “Reliability modeling of mesh storage area networks for internet of things,” IEEE Internet of Things Journal, vol. 4, no. 6, pp. 2047–2057, 2017.
[63] S. Zhou, K. Lin, J. Na, C. Chuang, and C. Shih, “Supporting service adaptation in fault tolerant internet of things,” in 2015 IEEE 8th International Conference on Service-Oriented Computing and Applications (SOCA), Oct 2015, pp. 65–72.
[64] D. Terry, “Toward a new approach to iot fault tolerance,” Computer, vol. 49, pp. 80–83, 08 2016.
[65] M. S. Ardekani, R. P. Singh, N. Agrawal, D. B. Terry, and R. O. Suminto, “Rivulet: A fault-tolerant platform for smarthome applications,” in Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference, ser. Middleware ’17. New York, NY, USA: Association for Computing Machinery, 2017, p. 41–54. [Online]. Available: https://doi.org/10.1145/3135974.3135988
[66] P. H. Su, C.-S. Shih, J. Y.-J. Hsu, K.-J. Lin, and Y.-C. Wang, “Decentralized fault tolerance mechanism for intelligent iot/m2m middleware,” in 2014 IEEE World Forum on Internet of Things (WF-IoT). IEEE, 2014, pp. 45–50.
[67] P. Stahl, J. Broberg, and B. Landfeldt, “Dynamic fault-tolerance and mobility provisioning for services on mobile cloud platforms,” in Proceedings - 5th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, MobileCloud 2017. Institute of Electrical and Electronics Engineers Inc., 06 2017, pp. 131–138. [Online]. Available: http://dx.doi.org/10.1109/MobileCloud.2017.7
[68] M. Nazari Cheraghlou, A. Khadem-Zadeh, and M. Haghparast, “A new hybrid fault tolerance approach for internet of things,” Electronics, vol. 8, no. 5, 2019. [Online]. Available: https://www.mdpi.com/20799292/8/5/518
[69] Y. Li, X. Sun, Y. Xia, P. Chen, Y. Li and Q. Peng, "M-MNFT: A Novel Modified (m, n)-Fault Tolerance Approach for Service Migration in Vehicular Edge Computing," 2023 IEEE International Conference on Software Services Engineering (SSE), Chicago, IL, USA, 2023, pp. 170-179, doi: 10.1109/SSE60056.2023.00031.
[70] H. Zhang et al., "Decentralized and Fault-Tolerant Task Offloading for Enabling Network Edge Intelligence," in IEEE Systems Journal, vol. 18, no. 2, pp. 1459-1470, June 2024, doi: 10.1109/JSYST.2024.3403696.