Review

Proactive Self-Healing Approaches in Mobile Edge Computing: A Systematic Literature Review

Olusola Adeniyi [1], Ali Safaa Sadiq [2], Prashant Pillai [1], Mohammed Adam Taheir [3] and Omprakash Kaiwartya [2,*]

1 School of Engineering, Computing and Mathematical Sciences, University of Wolverhampton, Wolverhampton WV1 1LY, UK
2 Department of Computer Science, Nottingham Trent University, Clifton Lane, Nottingham NG11 8NS, UK
3 Faculty of Technology Sciences, Zalingei University, Zalingei P.O. Box 6, Sudan
* Correspondence: omprakash.kaiwartya@ntu.ac.uk

Abstract: The widespread use of technology has made communication technology an indispensable part of daily life. However, the present cloud infrastructure is insufficient to meet the industry's growing demands, and multi-access edge computing (MEC) has emerged as a solution by providing real-time computation closer to the data source. Effective management of MEC is essential for providing high-quality services, and proactive self-healing is a promising approach that anticipates and executes remedial operations before faults occur. This paper aims to identify, evaluate, and synthesize studies related to proactive self-healing approaches in MEC environments. The authors conducted a systematic literature review (SLR) using four well-known digital libraries (IEEE Xplore, Web of Science, ProQuest, and Scopus) and one academic search engine (Google Scholar). The review retrieved 920 papers, and 116 primary studies were selected for in-depth analysis. The SLR results are categorized into edge resource management methods and self-healing methods and approaches in MEC. The paper highlights the challenges and open issues in MEC, such as offloading task decisions, resource allocation, and security issues, such as infrastructure and cyber attacks. Finally, the paper suggests future work based on the SLR findings.

Keywords: mobile edge computing; proactive self-healing; autonomous computing; fault tolerance; cyber security

check for updates

Citation: Adeniyi, O.; Sadiq, A.S.; Pillai, P.; Taheir, M.A.; Kaiwartya, O. Proactive Self-Healing Approaches in Mobile Edge Computing: A Systematic Literature Review. *Computers* **2023**, *12*, 63. https://doi.org/10.3390/computers12030063

Academic Editor: Paolo Bellavista

Received: 11 January 2023
Revised: 24 February 2023
Accepted: 9 March 2023
Published: 13 March 2023

1. Introduction

Mobile edge computing (MEC) is a distributed computing architecture that aims to bring computation and storage resources closer to the end users or devices at the network edge. With MEC, computing resources are made available at the edge of the network, typically within the radio access network (RAN), which reduces latency, improves network efficiency, and enhances the user experience.

MEC technology enables the development of new applications and services that require real-time data processing, such as augmented reality, autonomous driving, and smart cities. MEC also provides a more secure and cost-effective way to deploy applications and services by reducing the reliance on centralized cloud infrastructures.

Moreover, MEC allows for the creation of new business models by enabling service providers to offer value-added services that leverage local context information, such as location, proximity, and environmental data. MEC is expected to play a critical role in supporting the deployment of 5G networks, which will require new network architectures to enable low-latency, high-bandwidth, and high-reliability services.

MEC provides a low latency and high bandwidth environment, making it ideal for supporting a wide range of applications and technologies that require real-time data processing [1]. These include emerging technologies, such as IoT and 5G, as well as low-latency applications, such as connected vehicles, video analytics, smart cities, augmented reality, and bloackchain technologies [2]. Due to its ability to support these technologies and applications, MEC has become a highly significant and exciting topic in the field of computing and telecommunications.

Mobile edge computing is a complex and heterogeneous system, making it vulnerable to various faults and errors [3]. To address this, an effective network management approach must be implemented that can proactively handle faults before they occur. One such approach is self-healing, which enables a system to automatically detect and diagnose faults and take the necessary corrective actions to ensure system continuity and reliability.

To address the challenges in MEC, researchers have proposed various self-healing techniques. For instance, Nikmanesh et al. [4] utilized the discrete time Markov process to develop a decision-making analytical framework that can determine the optimal compensatory action in a failure scenario. Similarly, Porch et al. [5] proposed the use of a High-Order Recurrent Neural Network to detect performance degradation. In this approach, a series of user-reported reference signal received power (RSRP) readings are continuously collected and used as input data for the proposed model. The model is then trained to detect when performance degradation exceeds a certain threshold.

Due to the increasing complexity of the IT infrastructure, maintenance costs have been rising, as well as data growth, and the number of internet users has increased astronomically. Consequently, many researchers have focused on discovering answers through self-healing systems.

The term self-healing (SH) is closely associated with autonomous computing (AC), which was introduced at IBM's event in 2001. The event aimed to develop a system that could manage itself without human intervention to address the management issues arising from the growing computer systems and networks.

AC involves a system's capacity to control itself independently through various enabling technologies [6]. It is modelled after the human nervous system, which can regulate and control the body system. The goal of autonomous computing is to build self-running systems that can perform complex operations while hiding the system's sophistication from the user [7].

IBM developed AC in 2001 to create a distributed system on a large scale that is void of human intervention. The four essential features of AC are self-configuration, self-optimization, self-healing, and self-protection. The self-configuration feature focuses on configuring new system elements automatically during system deployments, upgrades, or after a system change without human involvement. Self-optimization allows the system to adapt automatically to increase efficiency and output during operation. Self-healing provides a system with the ability to automatically resume its activities when any malfunction happens. Self-protection enables a system to decide how to protect itself against malicious attacks [8].

AC has four key features, the first of which is self-configuration. This feature enables system elements to be configured automatically without human intervention during system deployment, upgrades, or after a system change has occurred. It encompasses the installation and configuration of new components, as well as the removal of damaged components and their automatic reconfiguration. Integrating large systems can be a challenging task, but self-configuration aims to ensure that an autonomic system can automatically configure itself based on high-level policies. This feature allows for a new component to integrate seamlessly into an existing system, with the rest of the system automatically adjusting to accommodate it.

The second important feature of AC is self-optimization, which enables the system to automatically adapt and improve efficiency and output during operation. By eliminating the need for manual configuration, this feature enhances the deployment process. Additionally, it performs regular system performance checks to ensure that key performance indicators (KPIs) remain within a reasonable threshold. In a complex system, optimizing certain parameters is crucial for achieving optimal performance, but improper adjustments can have unexpected consequences on the entire system. Self-optimization allows an autonomous system to continually seek ways to optimize its performance automatically.

The third feature of AC is self-healing, which equips a system with the ability to automatically resume its activities in the case of a malfunction. This feature involves a three-step process of detecting failures, diagnosing, and performing remedial operations. The primary objective of incorporating self-healing functionality is to enhance the stability and maintainability of the system. To achieve this, the self-healing process continuously monitors the system for faults and raises an alarm when necessary. Once an issue is detected, the system conducts a diagnosis to investigate the root cause of the problem. Then, appropriate remedial action is taken to address the issue and resume normal system activities. Overall, the self-healing feature of AC ensures that the system operates effectively and efficiently without human intervention.

Self-healing can be categorized into two approaches: proactive and reactive. The proactive approach anticipates the potential failures and takes action to prevent them from occurring, while the reactive approach responds to actual failures and takes action to restore the system to its optimal state. Proactive self-healing involves continuous monitoring of the system to detect potential issues and takes corrective measures to prevent them from developing into major problems. On the other hand, reactive self-healing responds to actual failures, detects the root cause of the problem, and applies the necessary remedial action to restore the system to its optimal functional state. Both proactive and reactive self-healing are crucial to enhancing system stability and reducing downtime.

Self-protection, the fourth feature, empowers a system to proactively protect itself against malicious attacks. It utilizes two key strategies to achieve this goal. The first strategy is to safeguard the entire system against potential attacks, and the second strategy is to anticipate problems based on reports and take necessary actions to avoid them. By using self-protection, the system can detect and mitigate the activities of foreign elements as they occur, preventing them from causing damage to the entire system. This feature offers high-level protection by identifying unusual activities and triggering the appropriate defensive actions. The self-protection process includes critical activities such as system monitoring, analysis, detection, and the mitigation of threats.

IBM proposes a reference model, called the MAPE-K (monitor, analyze, plan, execute, knowledge) loop, to implement autonomic control. This model consists of two main elements, the managed elements and the autonomic manager. The managed elements include sensors and effectors that interact with the external environment by gathering information and executing actions. On the other hand, the autonomic manager is responsible for controlling the system.

To achieve system control, the autonomic manager continuously monitors the entire system using information collected by the sensors. The monitoring component of the MAPE-K loop is responsible for gathering relevant environmental properties. Subsequently, the data is thoroughly analyzed to derive useful knowledge. Based on this knowledge, the planning component considers creating several modifications to the system. Finally, the execute component performs the execution of the appropriate modification to improve the system's overall performance.

In summary, the MAPE-K loop provides a structured approach for realizing autonomic control. By following this loop, the autonomic manager can monitor the system, analyze its performance, plan for improvements, and execute the necessary changes automatically. This approach ensures that the system remains stable, efficient, and responsive to its environment [6].

Efforts to mitigate failures have been ongoing, even before the emergence of self-healing systems. Traditional fault tolerance mechanisms, such as hardware and software redundancy methods have focused on failure detection and recovery. However, this approach requires human intervention, such as the use of experts to solve the problem, and often involves heavy hardware overhead. One of the key drawbacks of this approach is the time it takes to detect and rectify faults.

In addition, it is widely recognized that an expert's knowledge and experience in resolving a specific issue may not be documented or available for someone else to use, in case the same issue arises again. Moreover, recruiting and dispatching specialists to the failure site can be quite costly.

Given the limitations of traditional fault tolerance mechanisms, there is a growing need for a method that would enable systems to manage failures without human involvement. This is where autonomous systems come in, providing the capability for systems to function without human intervention.

High availability is crucial for business-critical IT systems that support operations that cannot tolerate downtime. When such systems experience downtime, the organisation faces not only financial losses, but also damage to its reputation. In certain domains, such as medical systems, high availability can mean the difference between life and death.

The use of an autonomous system to achieve high availability is critical because it can detect and resolve issues automatically. A system that can identify problems and self-heal without human intervention is more likely to achieve high availability. With self-healing, the system can recover automatically from failures, thus reducing downtime and ensuring that it remains available for users. Additionally, the ability of an autonomous system to monitor itself and detect problems before they become critical also contributes to high availability [9].

To achieve continuous availability, it is essential to have a system that can identify and address faults in real-time. An autonomous system with self-healing capabilities can provide this real-time response without human intervention. This feature ensures that the system continues to perform, even when one or more of its components fail. In addition to self-healing, self-optimization and self-protection features enhance the system's resilience by ensuring that the system remains in optimal condition and protected from external attacks. An autonomic control loop, such as the MAPE-K loop, can provide the necessary feedback and control required to keep the system in optimal condition. Therefore, the integration of autonomous capability can provide the necessary components for achieving high availability of an IT infrastructure system [10].

Indeed, achieving high availability is critical for the smooth operation of an enterprise. Downtime can lead to significant financial losses, impact customer satisfaction and reputation, and ultimately hinder business success. Therefore, it is essential to have a reliable and resilient IT infrastructure that can operate continuously without interruptions. However, ensuring high availability is a complex and challenging task, requiring the efficient detection, diagnosis, and remediation of failures. Autonomous systems can help achieve high availability by providing a self-healing mechanism that automatically detects and resolves faults, thus reducing downtime and improving the overall reliability of the system.

In this paper we focus on self-healing approaches in mobile edge computing, and we answer the following questions:

- How are SH approaches utilized in MEC?
- What design, theories, and algorithms could be used to support SH in MEC?
- What evaluation methods could be employed?

The rest of this paper is organized as follows. Section 1 provides introduction to MEC, while Section 2 describes the background. In Section 3 we present the conducted systematic literature review. Section 4 presents MEC challenges and use cases. Section 5 discusses autonomous approaches in MEC networks and related technologies. While in Section 6 we perform a review of self-healing approaches in MEC.

2. Background

The paradigm of mobile edge computing has recently garnered significant attention from both industry and academia, offering a promising solution to address the increasing performance demands of future applications. In this section, MEC is examined in detail, including related concepts and technologies, as well as the MEC architecture and network management strategies.

2.1. Mobile Edge Computing

Multi-access edge computing (MEC), previously known as mobile edge computing, is an architecture concept introduced by the European Telecommunications Standards Institute (ETSI) that offers cloud computing capabilities and IT services at the network's edge, be it a cellular network or any other network [11]. MEC combines information technology with telecommunications, providing application developers and content providers with access to the radio access network, which may not have been feasible before. The radio access network is a part of a mobile telecommunications system and implements radio access technology. By processing and storing content closer to cellular customers, MEC offers faster response times, improved application performance, and minimized network congestion.

MEC Related Concepts and Technologies

The MCE paradigm has gained significant interest in both industry and academia as a means of bringing computation, storage, and networking closer to the source of data generation. MEC is one implementation of the edge computing paradigm, which also includes fog computing and cloudlet. Fog computing is a decentralized computing architecture that deploys fog nodes across the network to perform computational tasks [12]. Similar to edge computing, fog computing delivers low latency, reduces network congestion, and brings computation and storage capabilities closer to end users. Fog computing and edge computing are similar in that they both offer these core functions, and the terms are sometimes used interchangeably. The primary difference between fog and edge computing is how data is handled or where the data processing occurs.

Fog computing processes data on fog nodes positioned in a local area network level, which may be a few network hops from the end user, while edge computing processes data on edge devices, such as smartphones, laptops, desktop computers, and tablets. In contrast, a cloudlet is a cluster of computers that is well-connected to the internet, providing computation and storage close to mobile devices [13]. Cloudlets appear similar to a "data center in a box" and can offer real-time interactive responses with low latency due to their proximity to the end user.

2.2. MEC Architecture

The MEC architecture allows for the deployment of services, applications, and content closer to the end user, which results in a low-latency and high-bandwidth environment. As shown in Figure 1, the MEC architecture comprises three distinct layers: the system layer, the host layer or server layer, and the network layer. The system layer includes a system-level management component that is positioned at the highest level of the entire system, providing a comprehensive view of the system. The host layer consists of the host component and the host layer management component, which are responsible for managing the services and applications deployed on the MEC server.

The host component of the MEC architecture includes the virtualization infrastructure, the mobile edge (ME) platform, and the ME applications. These components work together to enable the deployment of services, applications, and content closer to the end user, resulting in a low latency and high bandwidth environment. The network layer, which consists of the cellular network infrastructure, local networks, and external networks, plays a critical role in supporting the MEC architecture. Additionally, the network layer implements a radio access network (RAN), which is an essential part of a mobile telecommunication system. The RAN functions by transmitting signals to and from user equipment through the backhaul to the core network, facilitating the efficient transfer of data.

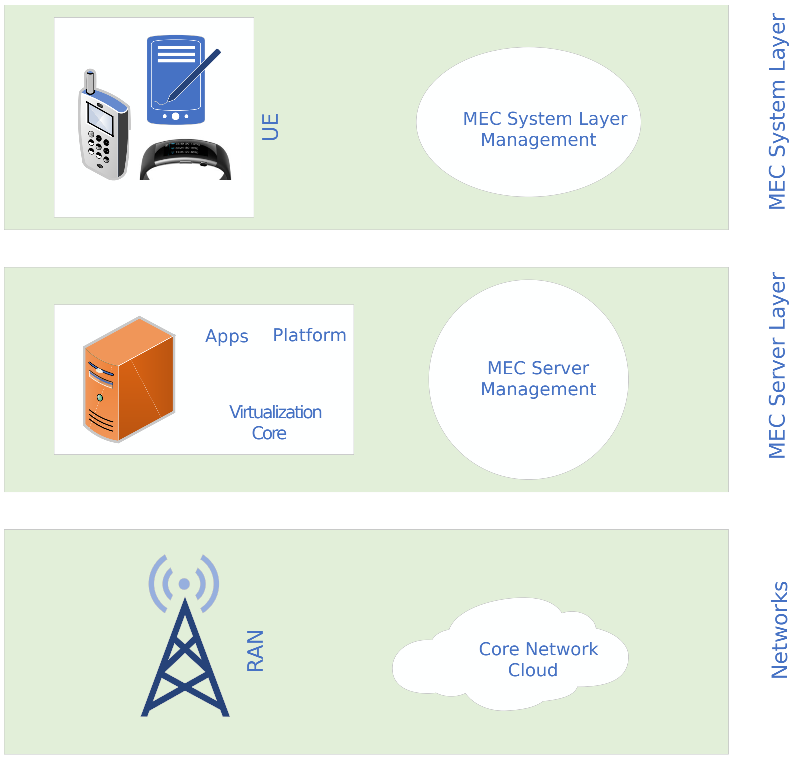


Figure 1. MEC architecture.

2.3. *Network Management*

As communication and computer networks continue to expand in size and complexity, managing these intricate infrastructures has become increasingly challenging. Effective network management is crucial for ensuring excellent quality of service (QoS) and monitoring, securing, and controlling access to the entire network.

Network operators must contend with a variety of issues to manage, maintain, and safeguard their network infrastructure. The problem of network management has received significant attention, and numerous solutions have been proposed. However, the difficulties of modifying the underlying infrastructure have made previous attempts to ease network administration a temporary fix. The closed proprietary and tightly integrated nature of network equipment limits the flexibility of the underlying infrastructure, thus constraining creativity and development.

Complexity of Network Management

The effective management of mobile network infrastructure and services in today's environment requires innovative approaches to address the added complexity brought about by virtualization and softwarization, while still reducing operating costs [14]. Network operators are challenged with executing complex policies and operations using limited device configuration commands. Configurations remain a critical aspect of network management, where high-level policies controlling the network depend on low-level configuration. However, low-level network rules are ill-equipped to handle rapid network changes and the demand for more functionality and complex regulations from operators continues to increase.

To address these challenges, the Zero Touch network and Service Management Industry Specification Group (ZSM ISG) was established by ETSI in 2017 to provide a framework for managing the complexities of network management across multiple domains. The ZSM ISG framework aims to automate all operational procedures and activities, including planning and design, delivery, deployment, provisioning, monitoring, and optimization. By providing a comprehensive framework for network management, the ZSM ISG seeks to enable autonomous management and overcome the difficulties associated with managing complex network infrastructure and services.

To achieve fully autonomous networks, artificial intelligence (AI), machine learning (ML), and big data analytics are crucial components. Recently, these methods have been used to intelligently perform various networking activities, such as administration, maintenance, and protection. For example, Fadlullah et al. [15] demonstrate the impressive results obtained from utilizing a deep-learning routing technique over a traditional routing strategy in a wireless mesh backbone network. Similarly, Kim et al. [16] utilize a graph neural network to address the issue of managing virtual networks and machines, which have increased in complexity. These studies showcase the potential of AI and ML in enhancing network management and support the notion that automation through these methods will be instrumental in realizing fully autonomous networks.

MEC, which integrates telecom, cloud computing, and IT services to bring computation, networking, and storage closer to the data source, has the potential to revolutionize the way subscribers access cloud services through the radio access network (RAN). However, due to its complexity and heterogeneity, MEC is susceptible to several faults, making a proactive self-healing system essential. Therefore, this systematic literature review aims to critically analyze the existing approaches, methods, and practices for achieving such a system in MEC, identify central issues and gaps in the current body of knowledge, and propose solutions.

To achieve this goal, the review presents a taxonomy of the topic, in terms of objectives, attributes, and approaches (as depicted in Figure 2). This taxonomy helps in understanding the different approaches used in the literature, identifying existing gaps, and proposing solutions. The review provides a foundation for the research by providing insight into previous studies and placing the research within the context of previous works on the subject.

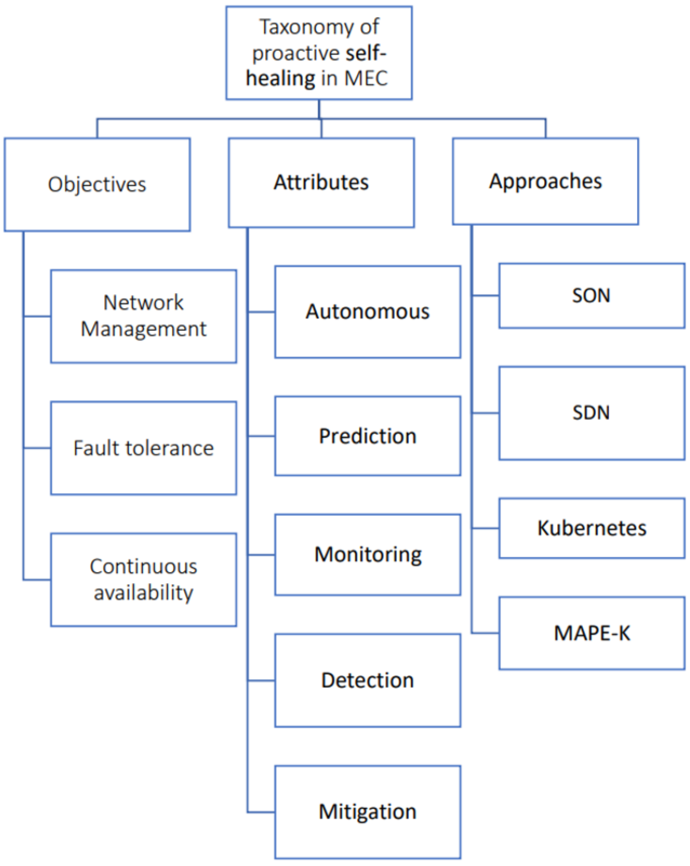


Figure 2. Taxonomy of proactive self-healing in MEC.

The review also describes the process followed in conducting the study, which involved formulating research questions, identifying keywords, and searching digital libraries, followed by evaluating the documents based on quality criteria and stating clear rules for inclusion/exclusion of the literature. The review synthesizes and provides a new viewpoint on the topic, identifies connections between theories and concepts, and discovers relevant factors pertinent to the topic. Overall, this review helps to advance the knowledge on self-healing systems in MEC and provides a basis for future research in this area.

Figure 3 illustrates the preferred reporting items for systematic reviews and meta-analyses (PRISMA) flowchart that outlines the systematic literature review process that has been used in this paper. This diagram serves as a visual representation of the various steps that were undertaken to ensure that our review was comprehensive, transparent, and reproducible. The process began with a digital library search using a pre-defined search string that yielded a large number of articles related to our topic. The articles were screened in several stages to eliminate duplicates, irrelevant studies, and studies that did not meet our inclusion criteria. The remaining articles were then subjected to a more thorough analysis, with the selected primary studies being used as the basis for our review. Our quality criteria were used to evaluate the studies and identify relevant information, which was then synthesized to answer our research questions. Through this process, we aimed to identify gaps in the existing literature and propose solutions for future research.

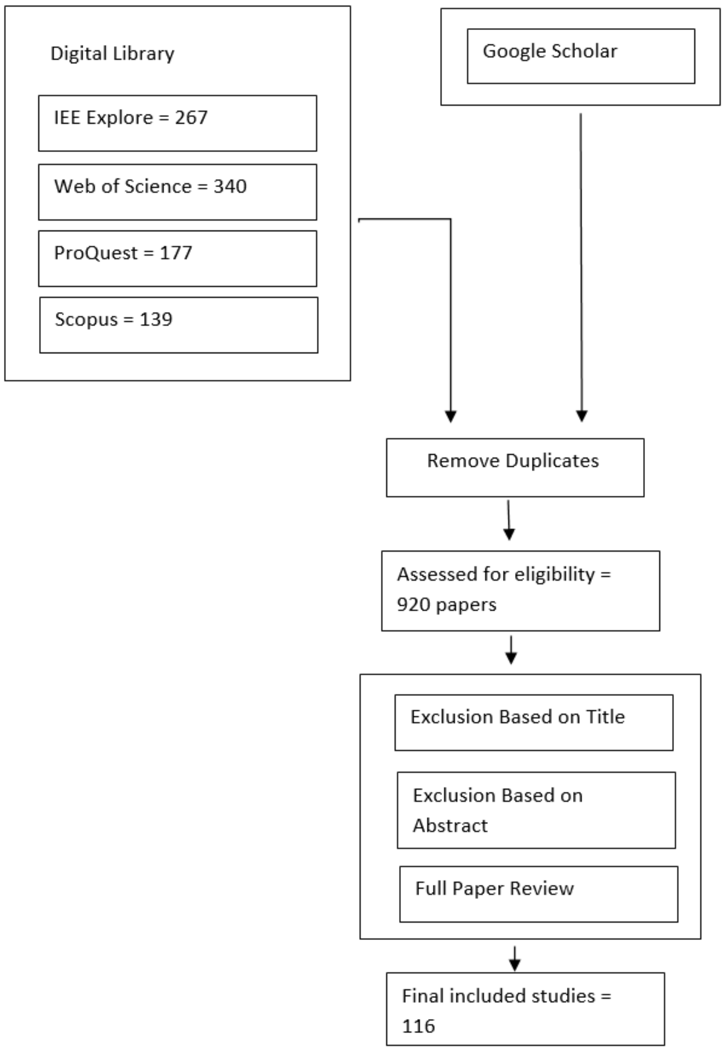


Figure 3. Systematic Literature Review (SLR) process.

3. Research Design

A well-designed research method and procedure are essential for conducting a comprehensive and rigorous SLR. In this section, we will provide details on the research design for our study.

Method: We conducted a systematic literature review of self-healing approaches in mobile edge computing. The method used in this review was guided by the PRISMA statement [17]. The review process involved searching for relevant literature, screening and selecting articles based on predefined inclusion and exclusion criteria, assessing the quality of the studies, and extracting relevant data.

Review Procedure: The review procedure consisted of the following steps:

1. Planning Stage: In this stage, we defined the scope of the review, formulated the research questions, and established the search strategy, inclusion and exclusion criteria, and quality assessment criteria.
2. Identification Stage: We conducted a systematic search of digital libraries, including IEEE Xplore, ACM Digital Library, ScienceDirect, and Springer Link, to identify relevant literature.
3. Screening Stage: We screened the identified literature based on the inclusion and exclusion criteria, which were established in the planning stage. In this stage, we removed duplicates and reviewed titles and abstracts to identify articles that met the inclusion criteria.
4. Eligibility Stage: In this stage, we assessed the full text of articles that passed the screening stage to ensure they met the inclusion criteria. We used a quality assessment tool to evaluate the quality of the selected studies.
5. Data Extraction Stage: We extracted relevant data from the selected studies and organized the data in a tabular format for further analysis.
6. Synthesis Stage: In this stage, we analyzed the extracted data and synthesized the findings to answer our research questions.

Selection Criteria: The inclusion criteria for the study were as follows:

- The study must focus on self-healing approaches in mobile edge computing.
- The study must be published in the English language and peer-reviewed.
- The study must be published between 2015 and 2022.

The exclusion criteria were as follows:

- Articles that do not focus on self-healing approaches in mobile edge computing.
- Articles that are not published in the English language or are not peer-reviewed.
- Articles that are published before 2015.

Adhering to the PRISMA statement and following a rigorous review procedure can help ensure the quality and validity of a systematic literature review. By applying specific inclusion and exclusion criteria, we can minimize the risk of bias and ensure that the studies included in our review are relevant and of high quality. This, in turn, can improve the reliability, replicability, and transparency of our review.

3.1. Research Methods

The SLR was conducted using the snowball sampling concept (see Figure 4). The idea, often referred to as a chain sampling or network sampling, starts with one or more study subjects. Following that, it continues based on referrals from the obtained papers at the initial stage out of the searching criteria. The cycle repeats until the desired sample is obtained or a saturation point is achieved [18].

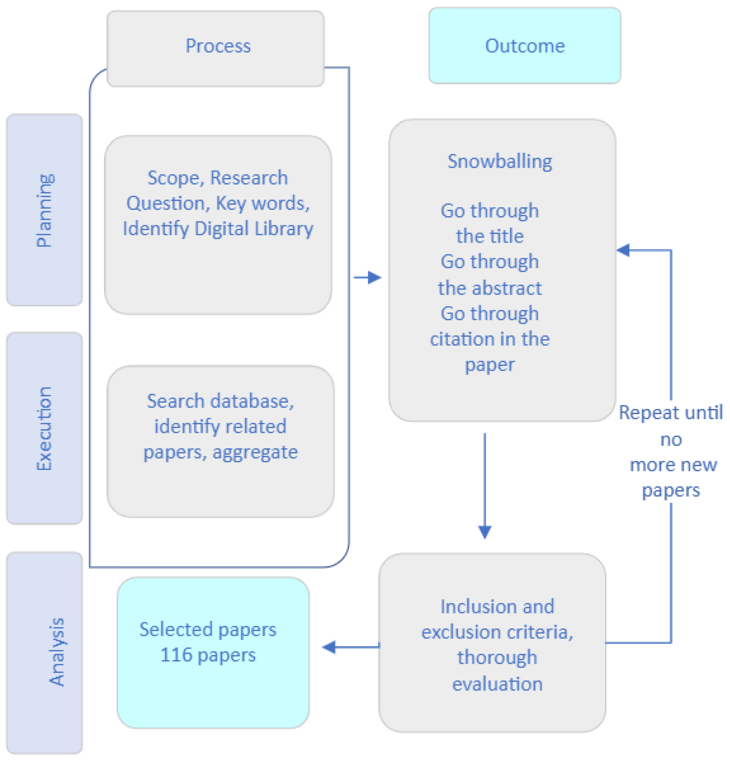

Process
Outcome
Planning
Execution
Analysis
Scope, Research Question, Key words, Identify Digital Library
Search database, identify related papers, aggregate
Snowballing
Go through the title
Go through the abstract
Go through citation in the paper
Repeat until no more new papers
Inclusion and exclusion criteria, thorough evaluation
Selected papers 116 papers

Figure 4. SLR-based snowballing procedure.

To contribute to the existing body of knowledge on self-healing approaches in mobile edge computing, we conducted a systematic mapping of the literature using the flowchart in Figure 5. This process allowed us to identify and highlight the different approaches used to address our research questions.

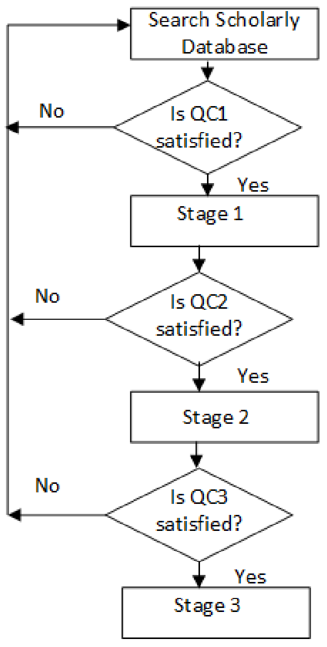


Figure 5. Quality criteria flowchart.

To ensure the relevance and comprehensiveness of our search, we carefully constructed our search string, taking into account the popularity and relevance of the keywords. We used Google Trends to measure the public interest in our keywords and to ensure that our search string was not limited in scope. This helped us to find a wide range of relevant literature to include in our review.

Figure 6 shows the popularity of the term "edge computing" over time, as measured by search interest. The numbers on the graph represent a score out of 100, with higher values indicating greater popularity. Over the ten-year period shown, the popularity of edge computing experienced an upward trend, with a low point in January 2012 and a peak in July 2021.

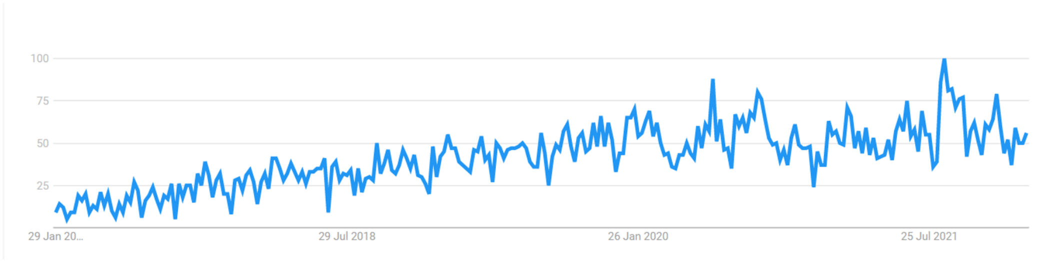


Figure 6. Edge Computing popularity between 2012–2022.

In Figure 7, we explore the popularity of the term "multi-access edge computing" (MEC), which is a newer name for mobile edge computing. The graph shows that MEC did not exist as a term before 2016, so search interest was low until that time. However, since then, the popularity of MEC has steadily increased, with 2021 marking its highest point yet.

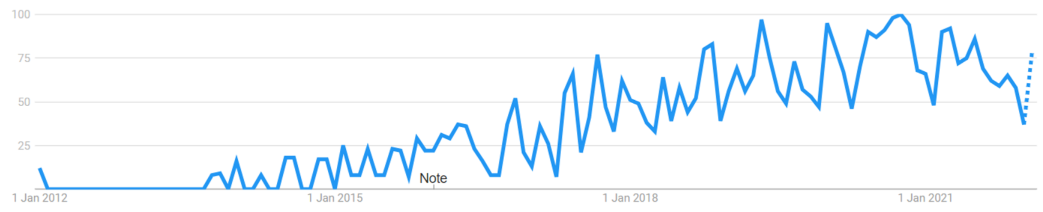


Figure 7. Multi-Access Edge Computing popularity between 2012–2022.

Figure 8 depicts the popularity of the terms "fault tolerance" and "self-healing" in the UK region. The graph reveals that the popularity of self-healing reached its highest value twice, once in 2013 and again in 2018, while the popularity of fault tolerance remained consistently lower, below 50.

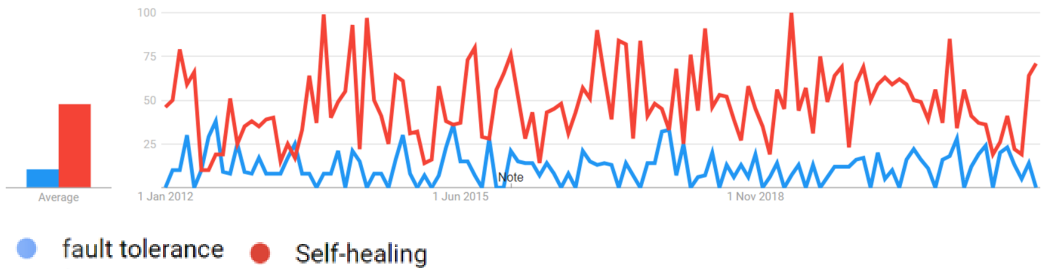


Figure 8. Fault Tolerance and SH popularity in UK.

Finally, Figure 9 compares the search interest in fault tolerance and self-healing around the world. The graph shows that both terms had similar levels of popularity, with neither exceeding a score of 50, until self-healing experienced a surge in popularity around 2021.

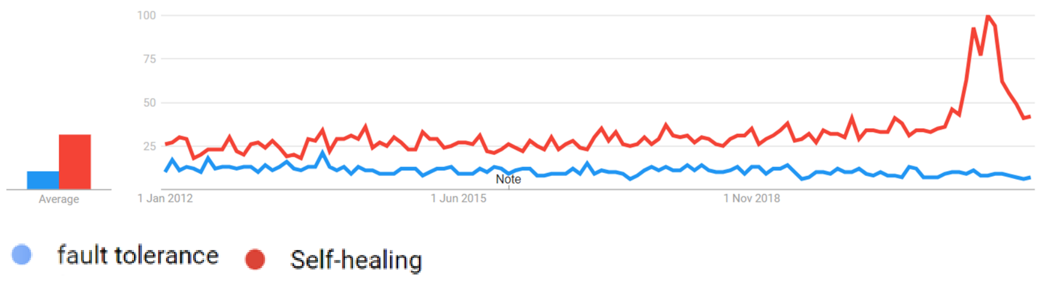


Figure 9. Fault Tolerance and SH popularity around the World.

We utilized several digital libraries to gather relevant materials related to our topic, as outlined in Table 1. The digital libraries we used include the IEEE Xplore Digital Library, Web of Science, ProQuest, and Scopus. IEEE Xplore is a comprehensive database that includes papers in the fields of engineering, computer science, and information technology. The Web of Science database records materials in the areas of social science, arts, and humanities. ProQuest, on the other hand, provides publications on medicine, surgery, and nursing sciences. Lastly, Scopus covers a wide range of subjects, including technology, natural sciences, information technologies, social sciences, and medicine, among others.

Table 1 lists the preliminary paper search conducted in four digital libraries.

Table 1. Search conducted in various digital library.

Source	Type	Url
IEEE Xplore	Digital library	https://ieeexplore.ieee.org/Xplore/home.jsp, (accessed on 10 November 2022)
Web of Science	Digital library	https://www.webofscience.com/wos/woscc/advanced-search, (accessed on 13 November 2022)
ProQuest	Digital library	https://www.proquest.com/index, (accessed on 17 November 2022)
Scopus	Digital library	https://www.elsevier.com/en-gb/solutions/scopus, (accessed on 21 November 2022)
Google Scholar	Search engine	https://scholar.google.co.uk, (accessed on 24 November 2022)

To achieve our set goal in searching databases, we utilized the search string shown in Figure 10.

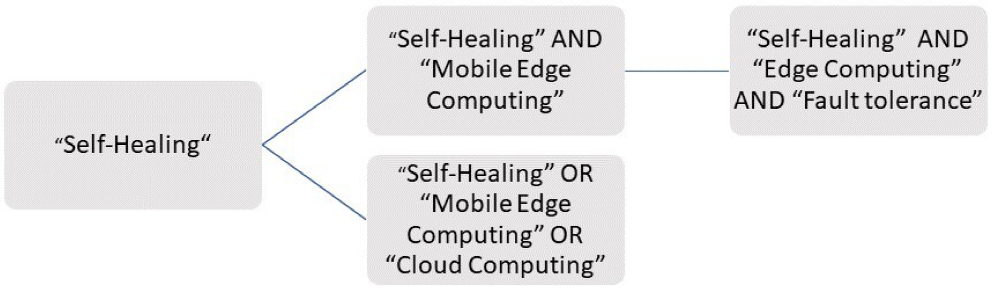

"Self-Healing"
"Self-Healing" AND "Mobile Edge Computing"
"Self-Healing" AND "Edge Computing" AND "Fault tolerance"
"Self-Healing" OR "Mobile Edge Computing" OR "Cloud Computing"

Figure 10. Search String.

Specifying "mobile edge computing" OR cloud computing OR edge computing was crucial, in order to minimize irrelevant application areas, such as SH for concrete materials, polymers, or SH in health care.

Inclusion and Exclusion Criteria: At this stage, we eliminate the studies that are not thematically relevant to the scope of this paper. The selection criteria we utilized include three stages.

We thoroughly examined the papers using a process that conforms with the following quality criteria (QC), as highlighted in Figure 5.

QC1. Is there an up-to-date study of the current knowledge?
QC2. Does the proposed study take a self-healing approach into cognizance or utilize any autonomic approach for network management?
QC3. Is there a comprehensive analysis of evaluations—are they explained in detail, are the study's objectives met?

Abstract Inclusion Check: In this stage, we reviewed the titles and abstracts of all the papers obtained from our search. We thoroughly scrutinized the abstracts and excluded some papers based on their content. For instance, if the abstract contained relevant keywords, we kept the paper for further processing. However, we did not evaluate the quality of the papers at this stage. The result of this stage helped us to distinguish between relevant and non-relevant papers.

After defining our scope, research questions, and keywords, we identified the digital libraries we would be using in our search.

Next, we conducted a search of the digital libraries using our search string. We then used QC1 to analyze the articles obtained. By reviewing the titles and abstracts, we identified articles related to our scope, and this was stage 1.

To achieve stage 2, we further analyzed the articles from stage 1 using QC2. We examined the content of these papers to ensure their relevance. At the end of this stage, we obtained thoroughly checked, relevant papers.

In stage 3, we further analyzed the papers from stage 2 using QC3. We looked at the methods and approaches used in the papers, the comprehensiveness of their analysis and evaluations, and whether the paper's objectives were met. The result of this stage led us to our primary papers.

Tables 2–5 are examples of the search conducted in four digital libraries. Showing the search string, the document type and source, as well as the year of publication.

Table 2. Examples of search conducted in IEEXplore digital library.

Search String	Title	Document Type and Year	Source Type
"Self-healing"	Multi-agent architecture for fault recovery in self-healing systems	Journal. 2021	Journal of Ambient Intelligence and Humanized Computing
"Self-healing" AND "Mobile Edge Computing"	Self-Healing on the Cloud: State-of-the-Art and Future Challenges	Conference. 2012	International Conference on the Quality of Information and Communications Technology
"Self-healing" AND "Edge Computing" AND "Fault tolerance"	A Multi-Agent-Based Self-Healing Framework Considering Fault Tolerance and Automatic Restoration for Distribution Networks	Journal. 2021	IEEE Access
"Self-healing" OR " Mobile Edge Computing" OR "Cloud computing"	Efficient Multi-User Computation Offloading for Mobile-Edge Cloud Computing	Journal. 2016	IEEE/ACM transactions on networking

Table 3. Examples of search conducted in Web of Science digital library.

Search String	Title	Document Type and Year	Source type
"Self-healing"	Self-healing systems—survey and synthesis	Journal. 2007	Decision support systems
"Self-healing" AND "Mobile Edge Computing"	Recovery for overloaded mobile edge computing	Journal. 2017	Future generation computer systems-the international journal of e-science

Table 3. *Cont.*

Search String	Title	Document Type and Year	Source Type
"Self-healing" AND "Edge Computing" AND "Fault tolerance"	Reliable and Fault-Tolerant IoT-Edge Architecture	Conference. 2018	IEEE sensors
"Self-healing" OR " Mobile Edge Computing" OR "Cloud computing"	Converging Mobile Edge Computing, Fog Computing, and IoT Quality Requirements	Conference. 2017	IEEE 5th international conference on future internet of things and cloud

Table 4. Examples of search conducted in ProQuest digital library.

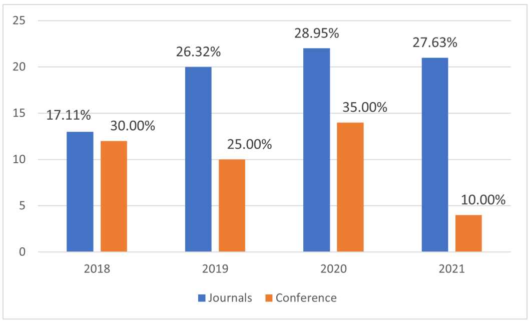
Search String	Title	Document Type and Year	Source Type
"Self-healing"	Modelling of intelligent distribution network self-healing control state transition based on FSM	Conference. 2019	IOP Conference Series: Materials Science and Engineering
"Self-healing" AND "Mobile Edge Computing"	An Efficient Nodes Failure Recovery Management Algorithm for Mobile Sensor Networks	Journal. 2020	Mathematical Problems in Engineering
"Self-healing" AND "Edge Computing" AND "Fault tolerance"	A fault-tolerant aware scheduling method for fog-cloud environments	Journal. 2019	PloS one
"Self-healing" OR " Mobile Edge Computing" OR "Cloud computing"	Self-Organized Cell Outage Detection Architecture and Approach for 5G H-CRAN	Journal. 2018	Wireless Communications and Mobile Computing

Table 5. Examples of search conducted in Scopus digital library.

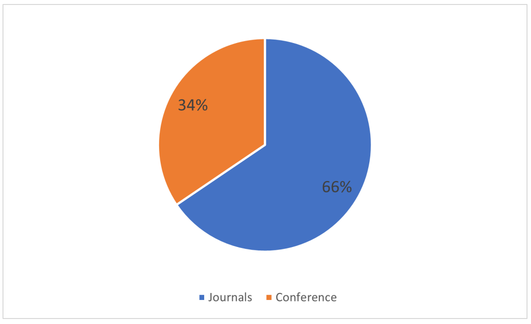
Search String	Title	Document Type and Year	Source Type
"Self-healing"	Anomaly Detection and Self-healing in Industrial Wireless Networks	Journal. 2020	Wireless Automation as an Enabler for the Next Industrial Revolution
"Self-healing" AND "Mobile Edge Computing"	Mobility-Aware Offloading and Resource Allocation in MEC-Enabled IoT Networks	Conference. 2020	16th International Conference on Mobility, Sensing and Networking
"Self-healing" AND "Edge Computing" AND "Fault tolerance"	A Game-Theoretical Approach for Task Offloading in Edge Computing	Conference. 2020	16th International Conference on Mobility, Sensing and Networking
"Self-healing" OR " Mobile Edge Computing" OR "Cloud computing"	An Efficient Task Offloading Strategy in Cloud-Edge Computing Under Deadline Constraints	Conference. 2020	IEEE 22nd International Conference on High Performance Computing and Communications

3.1.1. Results and Analysis

In this section, the outcome of the systematic review study are arranged in line with the research questions. As shown in Figure 11b, out of 116 articles investigated in this study, 34% came from conferences, while 66% were published in journals. Our search include papers from 2018–2021.



(**a**)



(**b**)

Figure 11. Publication outlet (**a**) year-wise percentage, (**b**) total percentage.

3.1.2. Discussion

This paper discusses the primary papers in two broad categories: management methods of edge resources and methods and approaches utilized in MEC for self-healing purposes.

Mobile Edge Computing Resource Management Techniques

The mobile edge computing environment has been the subject of a significant amount of literature, with a focus on resource management to optimize the quality of service (QoS). As the MEC environment interacts with highly non-stationary devices, task offloading to the MEC platforms has received much attention. To ensure that the QoS remains the same, the MEC architecture consists of several edge nodes. Hsieh et al. [19] addressed the challenge of optimizing task assignments in a system where diverse edge nodes process tasks together. The authors formulated the task assignment problem as a dynamic and random Markov decision process (MDP) and used online reinforcement learning to improve task processing execution, resulting in better performance than the commonly used standard technique. However, the impact of load balancing among the diverse edge nodes was not considered.

Similarly, Wu et al. [20] focused on end users' mobility by predicting the expected routes to be taken by mobile users using a deep learning algorithm. They then utilized an online mechanism that is dependent on QoS to resolve the offloading issue [21]. The authors claimed that their approach has a reduced offloading failure rate and a faster response time, compared to typical approaches. However, the prediction of users' mobility in real-life situations can be challenging, due to the complicated traffic network [22].

A considerable amount of the literature has been devoted to resource management in the MEC environment to optimize QoS. Given the interaction of the MEC environment with highly non-stationary devices, task offloading to MEC platforms has been the focus of much attention. The MEC architecture includes multiple edge nodes, and as mobile devices move, the QoS must remain consistent. Hsieh et al. [19] investigated the challenge of optimizing task assignments in a system where diverse edge nodes process tasks together. The authors framed the task assignment problem as a MDP and considered the problem as both dynamic and random. By using online reinforcement learning to improve task processing execution, they achieved a result better than the commonly used standard technique. However, their approach fails to consider the impact of load balancing among diverse edge nodes.

The network infrastructure is another crucial edge resource that significantly impacts QoS. Kan et al. [23] considered the use of both radio and computation resources to increase the performance of the MEC server. They approached resource allocation and offloading from the standpoint of cost minimization and claimed that their proposed solution significantly improved QoS by employing a heuristic approach to tackle the problem. However, their research was limited in scope and did not assess the impact of several base stations, as is the case in practice.

Zhang and Debroy [24] focused on the detrimental impact of wireless fluctuations on task execution time when real-time applications offload to the edge server. The authors offered a task offloading algorithm that can adapt to improve and balance energy consumption and total task completion time, claiming that their method provides better results. However, their paper lacks detailed information on the devices and parameters used, and there is no vivid comparison of their performance results with existing work.

Furthermore, Qiao et al. [25] focused on reducing energy usage when transmitting tasks to edge servers from a mobile device. The authors presented an energy-efficient algorithm that proved to be beneficial in lowering energy consumption. However, their research did not consider delays while offloading different tasks.

This category focuses on primary papers that use various methods and approaches in the literature for self-healing purposes. Our investigation shows that self-healing is an essential approach for next-generation network management solutions. Self-healing is required to manage both physical and virtual network infrastructures and services.

Some self-healing techniques gather information from MEC networks and perform diagnoses using various algorithms. For instance, Zoha et al. [26] presented a machine learning technique that gathers key performance indicators (KPIs) using minimize drive testing (MDT) characteristics. In an LTE network, the technique can automatically detect sleeping cells, providing a far better detection strategy.

In the context of self-organizing networks, Zoha et al. [27] addressed the detection and compensating approach for cell outages. The suggested detection component employed an approach based on minimum drive testing (MDT) functionality to gather data from mobile devices. Meanwhile, the second part of the approach, a compensating component, leveraged fuzzy-based reinforcement learning algorithms to deliver a suitable solution to users in the vicinity of the damaged cell. Specifically, the compensating component of their approach dynamically adjusts the antenna tilting and transmission power of neighboring cells to provide the best possible coverage to the affected area. The authors claimed that their findings demonstrate that the suggested framework is capable of autonomously detecting cell outages and compensating for them in a satisfactory way. The proposed approach can improve the network availability and performance by reducing service interruption and minimizing the negative impact on user experience.

Other research works have employed various algorithms for self-healing techniques, such as diagnosis. Khanafer et al. [28] used the Bayesian network for diagnosis in a UMTS network, while Ktari and Lavaux [29] proposed a Bayesian network model for system reliability and fault diagnosis. ADC [30] proposed a quadratic probing optimization algorithm for network node failure, and Khatib [31] utilized fuzzy logic as a supervised method for root cause analysis.

However, to effectively study the impact of faults on performance metrics, a diagnosis method requires a historical database of tagged fault instances. Typically, specialists do not focus on preserving the origin of the problem or the measures they took during troubleshooting activities. When there is an insufficient record of resolved instances, unsupervised techniques become the most suitable approach.

In an SDN environment, Vilchez et al. [32] proposed a self-healing approach for fault diagnosis based on Bayesian networks. However, the technique is reactive and does not prevent future occurrences of the same mistake. Thorat et al. [33] focused on switch-level link recovery and proposed a rapid recovery (RR) mechanism to quickly recover links. The method reduces backup flow rules for all the failing link's interrupted flows to a single flow rule.

4. Review of MEC Challenges and Use Cases

Mobile edge computing (MEC) has emerged as a promising solution to address some of the limitations associated with cloud computing. By facilitating the storage and computation of data closer to the source where it is generated, MEC has the potential to improve the performance of computing systems. However, despite its benefits, MEC faces several challenges that must be addressed to ensure its effectiveness.

One of the most critical challenges facing MEC is task offloading and resource allocation. The performance of the MEC system heavily relies on effective decision making regarding the offloading of tasks and allocation of resources. Given the large amount of data collected and generated by mobile devices within the MEC network, there is a need to determine the necessary information that must be sent to the edge server. In this regard, it is crucial to partition data into tasks before they can be offloaded to the server.

Furthermore, the limited resources available at the edge make it imperative to allocate them efficiently. To this end, proper resource allocation is crucial for ensuring the effectiveness of the MEC system.

A substantial body of literature has explored the challenges of task offloading and resource allocation in MEC. Notably, Zhong et al. [34] have contributed to this field by proposing a smart and effective task allocation system for large-scale MEC. The authors' model accounts for the MEC system's demand for minimal latency and reduced energy usage, which are crucial factors for ensuring high-performance MEC systems.

To address the task assignment problem, the authors proposed a nonlinear integer programming approach. This approach is based on a combination of Newton, linear search techniques, and conjugate gradients algorithms. By integrating these algorithms, the authors were able to achieve convergence, reduce complexity, and enhance scalability. This solution is particularly promising for addressing the resource constraints of MEC systems, while maintaining optimal performance.

Tong et al. [35] conducted research on task offloading in a MEC system with multiple servers and users. Their aim was to minimize server delay by finding the most suitable task offloading scheme for each computation task. To achieve this goal, the authors proposed two sub-optimal techniques that consider the harmony between computation power and assigned tasks.

In another study, Shan et al. [36] addressed the resource distribution problem in mobile edge computing. The authors proposed a sophisticated model that uses deep reinforcement learning to allocate resources effectively. The research showed that their approach led to reduced energy usage, lower latency, and a more balanced distribution of resources in the system.

Yang and Poellabauer [37] conducted a study on the handling of multiple task and service requests in mobile edge computing systems with limited computing resources. In their research, they proposed a technique for prioritizing computational tasks based on specific features and a reinforcement learning approach to solve the scheduling problem. To verify the feasibility and effectiveness of the approach, the authors conducted a simulation experiment on a MEC testbed. The study's main focus was to address the limitation of computing resources in MEC systems and how to optimally handle multiple tasks and service requests.

Shakarami et al. [38] proposed an autonomous computation offloading framework to tackle computational task offloading issues in MEC environments. To address the challenge of offloading decision making, the authors combined various techniques. The resulting framework yielded high accuracy in predicting energy usage, latency, and offloading, which are crucial metrics for task offloading decisions.

Naouri et al. [39] proposed a three-layer framework for addressing the task offloading challenge in MEC. The framework consists of three layers: the cloud layer, the cloudlet layer, and the device layer. To offload jobs that demand high processing power, the cloudlet layer and cloud layer are utilized. For operations with moderate computational requirements, but high communication costs, the device layer is used. By utilizing this approach, the transmission of large data to the cloud is avoided, and delays in processing data are significantly reduced.

To address offloading problems in large and diverse MEC networks with several cluster service nodes and various mobile task dependencies, Lu et al. [40] proposed a deep reinforcement learning (DRL) solution. The authors used iFogSim and Google Cluster Trace to simulate the task offloading challenge. By comparing their algorithm with the previous ones, the authors claimed that their new algorithm exhibited enhanced performance, in terms of latency, energy usage, load balancing, and average execution time.

Several studies have explored the use of caching as a potential solution to the challenge of resource management in mobile edge computing [41]. In particular, Safavat et al. [42] argued that a well-designed content caching strategy can play a critical role in reducing delays and improving the overall quality of experience. The authors also emphasize the importance of the content selection process, which involves both the caching and updating of content, as well as the expected duration for which the cached content remains relevant.

Similarly, Yang et al. [43] proposed an approach to efficiently allocate communication and computation resources through caching. The authors employ a long-short-term memory (LSTM) network to predict important tasks and formulate an optimization problem that involves task offloading, resource distribution, and caching choices. By leveraging the benefits of caching, the proposed approach can effectively reduce delays and improve overall system performance.

Li and Zhang's study [44] focused on the impact of latency on the quality of service in a mobile edge computing environment. The authors proposed a method for offloading tasks from mobile devices to edge and remote cloud servers, with the goal of minimizing the delay experienced during data transmission. The proposed method uses a combination of service caching and task offloading, resulting in reduced system delay and power consumption.

Similarly, Yu et al. [45] proposed an offloading scheme that integrates caching to reduce the delay experienced by mobile devices during task execution. The authors made the computational results available to multiple users to prevent repetition and improve efficiency.

Security is a significant challenge in mobile edge computing, which includes infrastructure security, cyber security, and other related issues. The infrastructure of mobile edge computing could be the target of malicious attacks, which could aim to manipulate data from different sources, in order to corrupt the outcome of the edge server's computation [46]. These attacks could include privacy attacks, data poisoning, and data evasion. The author advocates a multi-layer security approach, where the edge node can quickly and securely confirm that its peers before tasks are uploaded to address these issues.

Zhao et al. [47] discussed poisoning attacks that could be intentionally generated by malicious users to compromise the statistical results on the MEC server. The authors proposed a privacy-preserving scheme against poisoning (P3) to address this problem.

The vast amount of data gathered from various applications by MEC poses a new security threat, particularly during data access. To protect user data, access control is crucial, as emphasized by Hou et al. [48]. Meanwhile, Xiao et al. [49] addressed the need for a secure offloading strategy through a reinforcement learning algorithm. In addition, they suggested an authentication and cooperative caching strategy as a solution to various attacks aimed at mobile edge caching.

5. Autonomous Approaches in MEC Networks and Related Technologies

The concept of autonomic computing is a relatively new paradigm, and research has been conducted to propose new architectures, frameworks, and development processes.

In this section, we will examine the application of autonomous principles in managing mobile networks and related technologies. Additionally, we will review some existing architectures used to achieve autonomous functionalities.

5.1. Mobile Network Management with Autonomous Approaches

The heterogeneous nature of mobile edge computing has presented challenges in current network management approaches [50]. The complexity and diversity of the infrastructure and services provided make management increasingly difficult. As a result, there is a growing need for a network management approach that can consistently deliver high-quality services.

Traditionally, mobile networks rely on human experts to manually correct network faults. This reactive approach is inefficient and can negatively impact the quality of the experience. To address this, there has been a push for autonomous network management approaches that can automatically detect and remediate faults without human intervention.

The critical function of an autonomous system is its ability to operate without human intervention. An autonomous system (AS) is capable of performing computations on its own with little or no human involvement, as it is an evolving concept that is applicable in various fields, including maritime, space exploration, military, computing, manufacturing, robotics, and navigation [51].

An autonomous system (AS) is characterized by its hardware and software capabilities, which work together to identify a problem and execute a solution. It should also be able to gather information from external sources, process it, and make decisions accordingly, as it is a concept applicable in various fields, such as cloud computing, edge computing, and network management, which employ different approaches to achieve autonomous functionalities.

In computing technologies, such as cloud and edge computing, several methods are used to implement autonomous functionalities. Autonomous computing is one such approach. In network management, a self-organizing network (SON) is an approach that can be utilized to achieve autonomous functionalities.

5.2. Autonomous Approaches in MEC and Related Technologies (IoT and Bigdata)

Mobile edge computing is a critical enabler of IoT and big data technologies. By bringing computation and storage closer to the data sources, MEC accelerates the process of IoT data sharing and analytics. The autonomous decision-making principle is vital to enhancing the management of these technologies and services. It provides for real-time management without human intervention, making it essential for the Internet of Things (IoT), 5G, big data, mobile networks, and other related technologies.

IoT consists of various sensor devices, and the autonomic decision-making concept is essential for the installation and management of these devices. Manual consideration is ineffective, due to the large number of devices involved, making a dynamic and intelligent management technique necessary.

The autonomic control loop, consisting of autonomic managers and managed resources, can be introduced to the IoT component structure as long as the hierarchy remains. The components above serve as the autonomic manager, and the components below serve as managed resources. This approach can be applied to the components individually or collectively.

Autonomic decision making is critical for the effective management of IoT and big data technologies, which heavily rely on mobile edge computing to bring computation and storage closer to the data source. The real-time management provided by autonomic decision making without human intervention is particularly important in device management, access management, and identity management. By introducing the autonomic decision-making concept, the system can control and manage its functions automatically, resulting in greater effectiveness in all components.

Research initiatives have explored the fusion of IoT and autonomous control systems to create a responsive environment capable of adapting to changes. Lei et al. [52] proposed a system that utilizes sensors to gather data, servers to make decisions based on data analysis, and actuators to carry out those decisions. To achieve autonomy, the authors use reinforcement learning and deep reinforcement learning in the decision-making process. However, utilizing deep reinforcement learning for decision making can be affected by delayed control and incomplete perception.

To achieve autonomous functionalities in IoT, Wei et al. [53] proposed the use of broad reinforcement learning (BRL) to reduce traffic jams in smart cities. The proposed approach by Wei et al. highlighted the growing interest in using autonomous decision making in IoT management. By leveraging BRL, their solution aimed to reduce traffic congestion in smart cities by controlling traffic lights. The system's reinforcement learning algorithm learns the optimal timing and sequence of traffic lights based on real-time traffic data.

This approach demonstrates the potential of autonomous decision making in not only managing devices, but also in optimizing complex systems. With the increasing prevalence of IoT devices in smart cities, there is a growing need for autonomous management to ensure the efficient operation of these systems. Autonomous decision-making approaches, such as BRL, can be used in various applications, such as energy management, waste management, and public safety.

However, as with any autonomous system, there are also potential risks and challenges that need to be considered. The algorithm used in Wei et al.'s approach relies hvily on real-time traffic dataea. This raises questions about data privacy and security. In addition, there is a risk of the system being manipulated by malicious actors, which could lead to traffic accidents or other safety concerns.

To ensure the safe and effective implementation of autonomous decision making in IoT management, it is essential to address these potential risks and challenges. This includes developing robust security measures, ensuring data privacy, and implementing fail-safe mechanisms to prevent system failures. As research in this area continues to evolve, it will be crucial to strike a balance between the potential benefits and potential risks associated with autonomous decision making in IoT management.

Tahir et al. [54] have also proposed the use of autonomic computing for managing the IoT ecosystem with minimal human intervention. They describe the use of the MAPE (monitor-analyze-plan-execute) control loop for managing IoT components. The MAPE control loop assigns autonomic managers and managed resources to these components, enabling them to perform self-management and self-optimization without human intervention. This approach is particularly useful in scenarios where the number of devices and sensors is too large for manual management, and the system needs to adapt to changing conditions in real-time.

Big data is a field where autonomous functionalities are increasingly being applied to achieve greater efficiency in various tasks, including data security and privacy. One way in which autonomous decision making can improve security is through the automatic detection and prevention of intrusion. Traditional intrusion detection methods rely on sets of rules to analyze the source and behavior of an intrusion and apply preventive action, which can be slow and inefficient. This approach can result in a time gap between the detection and response, which can be crucial in preventing intrusions.

To overcome these limitations, there is a need to allow the system to protect itself without human intervention. This is where autonomous decision making can be applied to enhance the process of data security and privacy. By using machine learning algorithms to analyze data in real-time, an autonomous system can automatically detect potential intrusions and take preventive action, thereby reducing the response time and improving the overall effectiveness of the system. This approach is more efficient and reliable than traditional intrusion detection methods, making it a valuable tool for data security and privacy in the field of big data.

A number of studies have explored the use of autonomous functionalities in big data applications. One such study is that of Vieira et al. [55], which investigates the potential of integrating autonomic computing and big data techniques to build a system capable of processing large volumes of data from system operations to identify anomalies and formulate countermeasures in the event of an attack. The researchers propose a framework that uses machine learning algorithms to analyze data in real-time and make automatic decisions to protect the system. The framework is designed to be adaptive, so it can learn from new data and adjust its response accordingly. By automating the process of detecting and responding to threats, this framework provides a more effective way to secure big data systems, compared to traditional methods that rely on human intervention.

The autonomy intrusion detection system (AIRS) is a proposed method that is designed based on the MAPE-K model. One of its unique features is its ability to identify trends in massive log databases, which enables it to quickly detect and respond to cyber attacks. This is achieved by comparing signatures to suspicious actions, thus reducing the time lag between intrusion detection and response. The system's efficient response time makes it an effective tool for improving data security and preventing cyber attacks.

Kassimi et al. [56] also explored the use of autonomous systems for intrusion detection, specifically the discovery of anomaly data. Their system is designed to gather information about network traffic through a mobile agent, which then stores and analyzes the data. Using Hadoop infrastructure, the large amount of data collected is organized and analyzed by multi-agent systems to provide a self-healing intrusion detection system. This autonomous approach to intrusion detection allows for the efficient and effective identification of security threats, without relying on human intervention, thus reducing response time and improving the overall security of the system.

In a related research effort, Mokhtari et al. [57] developed an autonomous proactive task dropping mechanism that utilizes probabilistic analysis to achieve autonomous functionality in a distributed computing system. The main aim is to maintain the system's performance when faced with adverse situations. The authors employed a mathematical model to identify the most optimal task dropping decision that will enhance the system's stability, followed by a task dropping heuristic to achieve stability within a reasonable time frame. By adopting this approach, the system can continue to operate without human intervention and quickly adapt to changes in the environment, ensuring that it remains stable and performs optimally.

5.3. Review of Autonomous Architecture and Framework for MEC

The concept of autonomy enables a system to coordinate itself without external intervention, which can be particularly useful in a MEC environment. In this subsection, we will introduce several autonomous architectures and frameworks found in the literature that can be applied in MEC environments.

Biologically inspired frameworks and architectures—the development of autonomic computing has been influenced by the human nervous system. The human nervous system has the ability to control and regulate the body's systems, and this ability has been replicated in various autonomic computing approaches. These approaches focus on an entity that is responsible for controlling the systems, similar to how the brain controls the human body. By replicating the functions of the nervous system, autonomic computing can achieve self-regulation and self-optimization without external intervention.

There are various architectures and frameworks in the literature that can be useful in the context of MEC environments. Two of the relevant ones are biologically inspired frameworks and large-scale distributed system architecture.

Biologically inspired frameworks and architectures aim to replicate the human nervous system, which has the ability to regulate and control body systems without external intervention. In the context of AC, the approach is to create an entity responsible for controlling the system, similar to how the brain controls the human body.

Large-scale distributed system architecture includes applications such as Ocean store SMARTS from IBM and Auto Admin from Microsoft, which have been developed to manage distributed systems and databases automatically. These applications are designed to achieve the goals of AC, such as self-management and self-healing.

Another architecture that has been widely used to achieve autonomous behavior is the agent-based architecture. In this type of architecture, agents interact with each other to create a system that can oversee itself automatically. The key aspect of this architecture is the agent, which is an autonomous entity that can function without the need for a central control system.

The component-based architecture places emphasis on autonomic components and their interactions, with the goal of achieving a set objective. These components carry out monitoring, analysis, planning, and execution, which are the core functionalities of autonomous computing. By breaking down the system into smaller, more manageable components, this architecture enables a more efficient and effective approach to achieving autonomy. The autonomic components interact with each other, exchanging information and coordinating their actions to achieve the overall objective.

Technique-focused architecture is an AC architecture that leverages AI and control theory to optimize system performance. Its primary objective is to generate actions, acquire feedback, manage actions by retrying or re-planning, and generate alternative approaches to achieve specified goals. The architecture comprises features that enable learning from the environment, taking action, receiving feedback, planning, and re-planning, thus achieving autonomous behavior in a system.

On the other hand, service-oriented architecture focuses on the interaction of services to achieve autonomous functionalities. Services are loosely coupled and operate independently, offering interoperability and scalability. This architecture enables the creation of self-managed systems through the dynamic configuration of services, allowing for fault tolerance and self-healing capabilities.

5.4. Discussion

The process of automatically identifying faults is an essential aspect of a self-healing solution. However, there is limited research on self-healing in the mobile network domain. This can be attributed to several factors. Firstly, it is challenging to determine the most prevalent faults and their warning signals in mobile telecommunication networks, due to their complex nature. Moreover, experts who are responsible for identifying and solving faults often do not document them. Secondly, obtaining historical data from live networks is difficult. As a result, most self-healing solutions in existing studies are based on modeling faults in simulators. However, testing such solutions on a live network can slow down the system, which is counter-productive, making it challenging to evaluate their performance.

6. Review of Self-Healing Approaches in MEC

Self-healing capabilities integrated into network management systems hold significant potential for achieving crucial goals, such as maintaining QoS, while simultaneously reducing capital investment, operational expenses, and maintenance costs. Self-healing systems offer a methodology for identifying, categorizing, and mitigating faults. In this section, we will explore the various implementations and strategies utilized in achieving self-healing capabilities.

6.1. Various SH Implementation in MEC

MEC is a hybrid of various components that combines IT services with telecommunications. At its core, MEC involves deploying MEC servers within mobile networks. These servers act as virtualization infrastructure that hosts MEC services, software-defined networking, and network function virtualization. This provides a flexible environment where service providers can deploy their software across MEC systems from multiple vendors, creating a more open and interoperable ecosystem.

6.1.1. Network Functions Virtualization (NFV)

In a network, nodes serve as connection points and are responsible for receiving, generating, storing, and transferring data along network routes [7]. These nodes can recognize, process, and forward transmissions to other network nodes. Network nodes can also perform functions such as load balancing, firewalls, intrusion detection systems (IDS), and intrusion prevention systems (IPS). However, these network functions have traditionally required specialized hardware, which can be expensive. To address this, the concept of network functions virtualization (NFV) has emerged, which turns these network functions into software applications that can run on a virtual machine, eliminating the hardware cost constraint. NFV provides a flexible, agile, and scalable solution for networks [58].

6.1.2. Software Defined Networking (SDN)

The traditional approach to transferring and managing data in a network relies on the hardware and software integrated into devices such as routers and switches. However, SDN technology separates the software responsible for directing data flow and network management from the networking device that actually moves the data [59]. This allows for greater flexibility and control over network behavior, as well as easier management of network traffic and security policies.

SDN is a network architecture that aims to replace proprietary hardware-based equipment and services with common, software-driven methods [60]. This approach provides several benefits, including increased security, improved network visibility and control, and the ability to quickly adapt to changing enterprise requirements. With SDN, the software that controls how data travels through the network is separated from the underlying hardware responsible for transferring the data [59]. This coordination enables organizations to manage their entire network more intelligently, including in multi-location and multi-connectivity technology environments. By utilizing a unified network management software, businesses can improve flexibility and better position themselves to adapt to future needs.

The MEC platform is built on the principles of distributed systems, which consist of multiple autonomous components that work together to appear as a single system to users [61]. However, since distributed systems are prone to multiple points of failure, self-healing capabilities are necessary for the system to recover from failures on its own. Therefore, various approaches are used to achieve self-healing capabilities in MEC systems.

MEC environments can experience different types of failures, which can be broadly categorized as network infrastructure failure and MEC system failure. Network infrastructure failure may result from hardware failures or unstable connections within the mobile network. MEC system failure can be caused by a range of issues, such as hardware breakdown, software breakdown, or an overloaded MEC resource.

The MEC framework comprises the MEC host, which includes the MEC platform, MEC applications, and virtualization infrastructure. The MEC host serves as the server where MEC applications run together with the virtualization infrastructure. As this component plays a crucial role, achieving self-healing capability is imperative to ensure its uninterrupted operation.

For a business application to maintain a positive reputation in the market, it must run continuously without interruptions due to technical errors, feature updates, or natural disasters. Achieving a continuous application process can be challenging in today's complex, multi-layered infrastructure. This is where self-healing technology comes in, enabling applications to function seamlessly, even in the face of such challenges.

6.1.3. Self Organizing Network (SON) Approach

To achieve self-healing in the Network infrastructure component of MEC, it's essential to have autonomous functionality, and one approach that can be useful is the self-organizing network (SON) paradigm. A SON is a highly effective approach to enhance network management and meet the demand for high-quality services.

In order to achieve self-healing in the network component of a MEC system, the network must be both programmable and autonomous, as noted by Santos [62]. One effective paradigm for achieving this is the self-organizing network (SON), which is a technology designed to enhance the management and optimization of mobile radio access networks. By offering automated processes for network design, installation, management, and healing, SON has the potential to reduce the need for human involvement, as Ramiro has pointed out [63].

The management and operation of modern radio access networks (RANs) can be extremely complex, partly due to the heterogeneity of the deployed technologies and their lack of interoperability. In the radio access domain, the SON represents a promising approach to addressing these challenges, as they help to eliminate human error, reduce repetitive automation, and streamline tedious processes. By leveraging autonomic mechanisms, which can be either software or hardware-based, SON has the potential to significantly improve responsiveness and computational capacity, outpacing the capabilities of human experts and highlighting the need for self-management in RANs.

SON comprise a variety of processes that enable configuration, optimization, and healing in a self-regulating manner within a mobile network. By lowering capital expenditure (CAPEX) and operating expenditure (OPEX), SON has become a critical necessity in the operation of next-generation mobile networks.

SON functionalities have simplified the maintenance of communication networks that would otherwise be difficult and time-consuming. The programmability, flexibility, and dynamic nature of SON features have significantly enhanced the network's operation by reducing human error and allowed for greater automation.

As previously mentioned, the SON function is traditionally divided into three categories: self-configuration, self-optimization, and self-healing. Self-healing (SH) was created as a feature of SON in compliance with the 3GPP standard to address three aspects: cell outage, self-recovery of network element (NE) software, and self-healing of board problems. Initially, the industry's focus was on cell outage management, which is addressed by two primary components of SH: cell outage detection (COD) and compensation.

COD is responsible for automatically identifying cell outages, including sleeping cells and degrading cells. The compensation component provides an effective strategy for rectifying the identified flaws. The introduction of SH functionalities has significantly simplified the manual maintenance of communication networks, which would otherwise be difficult and time-consuming. By enabling the automatic identification and remediation of faults, SH features make the network more flexible, dynamic, and programmable, while reducing capital and operational expenditures in the operation of next-generation mobile networks.

The goal of cell outage management is to maintain network performance by continuously monitoring cells for outages and automatically adjusting network settings to restore optimal performance. In the compensation component, the algorithm adjusts the cell transmit power, antenna tilt, and azimuth to compensate for the outage [64].

In the self-healing process, the monitoring module detects a failure or a network device that triggers an alarm. Further analysis is performed to identify the root cause of the defect, using information from previous experience, measurements, or testing results. Once the defect is identified, the compensation module initiates the necessary repairs autonomously.

Considerable research has been dedicated to addressing the COD component of self-healing in a SON environment. One proposed solution to detecting cell outages is the use of hidden Markov models. This system automatically collects information about the current status of base stations (BS) and uses that information to predict when an outage may occur. While the method has been proven to predict BS conditions with 80% accuracy and a cell outage detection rate of 95%, it does not account for the influence of time or the historical sequence of cell states, which may provide useful insights for identifying erroneous cell profiles [65].

In the study by Yu et al. [66], a two-step architecture was proposed for cell outage detection, which includes data collection and data analysis phases. The first step focused on aggregating data from different sources, while the second step analyzed the data received in the first phase. The authors employed a modified LOF (Local Outlier Factor) detection technique to identify outages in various cell types and evaluate their approach using a use case. However, the study was limited in scope, as it did not consider resource allocation, load balancing, or the diagnosis and compensation components necessary for self-healing.

In contrast, Palacios et al. [67] have explored the use of statistical models to diagnose network problems, conduct recovery operations, and quickly compensate for the issue with the help of surrounding cells.

Mobile network operators have increasingly deployed more sophisticated networks, leading to the challenge of maintaining and managing these complex networks. To address this challenge, SON have become critical for automating network operations. However, current SON solutions may not be fully utilizing the massive amounts of network data available. Leveraging this data can enable the development of a proactive system that learns, predicts, and adapts dynamically to changing network requirements and volatile conditions.

The integration of SDN and NFV with SON can deliver an intelligent network. By separating the control plane from the data plane, SDN allows for more flexible network control and management, while NFV virtualizes hardware such as switches and routers. SON then acts as the "brain" of the network, reorganizing the network as needed to optimize performance. These technologies enhance SON's ability to deal with dynamic networks by making the network programmable and adaptable to changing conditions.

6.1.4. Kubernetes Approach

Kubernetes is a self-healing solution that can be leveraged on an MEC platform to deploy and maintain uninterrupted services and applications in the face of technical glitches, feature updates, or natural disasters.

Released in 2015, Kubernetes is now one of the most widely used orchestration systems. It offers a unified platform for deploying applications and abstracts the underlying infrastructure, enabling developers to deploy applications without the need for support from system administrators. Meanwhile, system administrators can monitor and manage the availability of the supporting infrastructure. The primary objective of Kubernetes is to simplify deployments, saving time and effort [68]. To achieve this, Kubernetes provides a wide range of functionalities, including:

- Automatic bin packing: To maximize usage, the automatic bin packing feature of Kubernetes arranges containers based on resource demands and constraints.
- Horizontal scaling: This feature adjusts the number of executions and resource based on various requirements. It ensures the workloads are shared without affecting the existing ones currently running.
- Secret management: Kubernetes dissociates the management of application configuration information and sensitive data from the container image.
- Self-healing: Kubernetes has a unique self-healing feature that may restart failed containers, replace containers, and destroy containers that do not react to a user-defined health check [69]. Additionally, Kubernetes employs its self-healing functionality in a failed node by replacing and redistributing the failed containers to other active nodes.

Auto-scaling is one of the most critical capabilities of Kubernetes in the MEC environment, as it allows containerized applications and services to operate reliably without human intervention. With auto-scaling, services are rendered based on the request from the users, enabling high availability and scalability.

Numerous research studies have explored the use of Kubernetes in MEC. For instance, Ju et al. [70] proposed an approach that predicts the incoming range of workloads to scale up and down as necessary. The approach allows users to customize metrics, forecasting models, and scaling strategies to better suit the requirements of their applications. However, the approach is not comprehensive enough as the process of adjusting metrics for auto-scaling to take place is manual.

Additionally, Casalicchio et al. [71] conducted a study on the behavior of the Kubernetes horizontal pod auto-scaling algorithm and developed customized auto-scaling algorithms that depend on accurate metrics to make scaling decisions. However, their approach did not take into account relative metrics when dealing with a wide range of workloads and, instead, relied solely on precise metrics for scaling choices.

6.1.5. MAPE-K Model with Case Based Reasoning (CBR) Approach

The MAPE-K model is a self-healing infrastructure that can be integrated with an existing software system to provide self-healing ability. This infrastructure acts as an autonomic manager to control the managed element, which, in this case, is the software system. By using this model, the autonomic manager can monitor the software system and detect faults or anomalies in real-time. Once a fault is detected, the autonomic manager can then take appropriate actions to resolve the issue and ensure that the software system continues to function correctly. The MAPE-K model consists of four components: monitoring, analysis, planning, and execution, which work together to provide a self-healing ability to the software system.

The importance of uninterrupted service cannot be overstated in large-scale software systems. Service failures are an expected occurrence and can be caused by a range of factors, including hardware failure, software bugs, network issues, and even cyber attacks. As a result, it is essential to introduce appropriate self-healing methods to ensure that the system can quickly recover from such incidents and continue to provide the desired service.

Self-healing is the ability of a system to detect, diagnose, and repair faults autonomously without human intervention. A self-healing infrastructure can be integrated with an existing software system to provide this ability. The self-healing infrastructure serves as an autonomic manager that controls the managed element, which is the software system. The autonomic manager detects faults, diagnoses the cause of the fault, and takes corrective actions to restore the system's normal operation.

Developing a self-healing software system requires a clear understanding of the types of service failure that may occur, the diagnosis of the fault, and a repair strategy. A self-healing approach should be able to treat diverse malfunctioning scenarios in the software system effectively.

One effective problem-solving approach for building self-healing systems is case-based reasoning (CBR). CBR is a problem-solving approach that draws on the knowledge of previously encountered events, referred to as cases. It works by obtaining previous cases that are similar to the present one and reusing previous successful solutions. CBR enables the creation of a knowledge base of previous cases that can be applied to the current situation. The use of CBR is well suited in fields where structured and generally recognized background information is not accessible.

6.1.6. Multi Agent Approach

Multi-agent systems (MASs) are a popular approach for tackling complex problems by dividing them into smaller pieces. In this technique, each agent is an independent entity that detects and collects information from its environment to achieve a specific objective. According to Zhou et al. [72], an agent acts autonomously on this information and chooses how to accomplish its tasks.

The MASs approach assigns specific pieces of a complex problem to each agent, based on their objective, communication with neighboring agents, and history of previous activity. The agent then processes its portion of the problem based on these variables. This division of labor allows for more efficient and effective problem-solving, compared to centralized systems, which can become overwhelmed by the complexity of the problem.

Multi-agent systems (MASs) have emerged as a promising technique for tackling complex problems by breaking them down into smaller parts that can be handled by individual agents. An agent is an entity that perceives and collects information from its environment, acts autonomously based on this information to achieve specific objectives, and interacts with other agents to coordinate their actions [72].

The flexibility of MASs makes them applicable in various fields. For example, Zhou et al. [72] applied multi-agent reinforcement learning to solve the task offloading and resource allocation problem in 5G and Wi-Fi 6 networks. To address the problem, they proposed a two-part method. In the first part, they used a Lyapunov optimization-based approach to select job scheduling and processing power. The second part involved an online multi-agent reinforcement learning component and a game theory-based method to select offloading links and determine power transfer.

Heydari et al. [73] proposed a multi-agent technique to tackle the resource allocation problem in a scenario with multiple mobile devices and edge servers. To address the problem, they converted it into a Markov decision process (MDP) and utilized an actor-critic approach with neural network function approximation to determine the best offloading strategy. By employing this approach, the authors demonstrated the effectiveness of multi-agent systems in solving resource allocation problems in edge computing.

6.2. *Self-Healing through Software Defined Network*

Effective network management is crucial for organizations to achieve smooth business operations. However, it can come at a significant cost, both in terms of capital and operational expenditure. To reduce expenses, organizations are always searching for more efficient ways to manage their networks. This is where software-defined networking (SDN) comes in—it is a network management paradigm that provides simplicity and innovation by abstracting the network, making it controllable from a single location with specialized software.

SDN splits the network control and the data plane, enabling easier network design, implementation, and management. It is a widely used approach that provides a sophisticated way of applying self-healing systems in a network. With SDN, applications can be deployed quickly and network resources can be utilized to their fullest potential. By separating the control plane from the data plane, SDN enhances the network's ability to adapt to new, different, or changing requirements, while also offering centralized management and control of the network.

SDN is composed of three distinct layers: the application layer, control layer, and data layer. Each layer is responsible for carrying out specific functions that collectively provide a dependable and scalable network. The controller is a major component that ties these layers together. It is responsible for managing all operations and enforcing network policies.

The controller plays a critical role in the implementation of intelligent networks. It manages flow control to the networking devices, as well as applications and business logic. There are various controller platforms available, including Floodlight, Beacon, and OpenDayLight. These platforms provide a centralized point of control for managing network operations and enabling organizations to customize their networks according to their specific needs.

The SDN controller plays a central role in managing network equipment. It is composed of various components that execute different network functions, such as maintaining a list of network devices and their capabilities, monitoring network activity, and computing network statistics.

To communicate with network devices such as routers and switches, SDN controllers typically use protocols such as OpenFlow and the Open vSwitch Database (OVSDB). Table 6 summarizes different protocols used in SDN and their limitations. OpenFlow is a widely used protocol that defines how a controller interacts with network devices and provides a standardized approach to traffic management. With OpenFlow, all packet-moving decisions are centralized in one location, so network configuration can be performed across the entire network, regardless of the specific switches being used.

Table 6. Protocols used by SDN.

Name	Function	Protocol Type	Limitations
Openflow	Used for controlling traffic from a centralized location	Network protocol	Table sizes are limited, and events are processed in a controlled manner.
Border Gateway Protocol (BGP)	Used for exchanging routing information between gateway hosts in a network of autonomous systems	Distance vector routing protocol	Issues arising from the delay of BGP convergence time
NETCONF	It offers a safe method for configuring a firewall, switch, router, or other network equipment for an administrator or network engineer	Network Management protocol	Utilises YANG models, however there is presently no industry-wide support for a uniform set of YANG data models. Programmes must request and set data on a variety of devices.
Extensible Messaging and Presence Protocol (XMPP)	It is used for instant messaging and detecting online presence. Using the protocol, servers can communicate almost in real-time	XML elements streaming protocol	Prone to various security issues. The protocol require additional functionalities for protection
Open vSwitch Database Management Protocol (OVSDB)	An OpenFlow configuration protocol designed to manage Open vSwitch implementations	Management protocol	OVSDB will avoid committing the entire transaction's configuration, even those that are properly configured, if there is a single configuration error.
MPLS Transport Profile (MPLS-TP)	It is a transport profile for multiprotocol label switching. Transport networks use MPLS-TP as a network layer technology	Transport protocol	MPLS is perfect solely for point-to-point connection, it is however not a suitable choice for operations in the cloud

OpenFlow relies on a set of messages that the controller and switches use to communicate with each other. The controller uses these messages to program the switches and control the flow of user traffic.

As previously mentioned, OpenFlow is the most widely used protocol for network topology discovery in SDN. However, the protocol is not suitable for production deployment due to its security vulnerabilities. OpenFlow utilizes a non-secure link layer discovery protocol (LLDP) frame structure to discover the links between switches. The problem is that the LLDP packets sent to the controller by switches during the link discovery procedure are not authenticated, which leaves the network vulnerable to various attacks, such as switch spoofing, link fabrication, and LLDP flooding.

To address this vulnerability, Alharbi et al. [74] examined the topology discovery process in SDN and proposed a method of including a keyed-hash-based message authentication code (HMAC) in each LLDP packet. This provides both authentication and packet integrity, preventing the compromise of the topology information.

Similarly, Azzouni et al. [75] demonstrated the security and performance issues with OpenFlow and proposed the secure and efficient OpenFlow discovery protocol (SOFTDP) as a solution. The study introduced minor adjustments to the OpenFlow switch design, enabling it to detect link events and notify the controller. Additionally, bidirectional forwarding detection, dropped LLDP rules, and hashed LLDP content were introduced to enhance security and performance.

6.3. Review of Relevant Self-Healing Methods in MEC

Self-healing is a critical feature of autonomous systems that enables automatic fault detection, diagnosis, and remedial actions. However, compared to self-configuration and self-optimization, less research has been conducted on self-healing in mobile networks. This is because effective self-healing requires knowledge of past network faults, which may not be readily available to researchers. Additionally, identifying the symptoms of different network faults and their root causes often relies on expert knowledge that may not be documented.

Figure 12 depicts the components of a self-healing framework, which comprises three main parts: the detection component, the diagnosis component, and the compensation component. The self-healing process begins with the monitoring component, which scans the network to collect information about its health status. The collected data is stored in a database and transmitted to the next component. The fault detection component processes the received information, and if it detects any fault, it is forwarded to the fault diagnosis component. The fault diagnosis component consults the knowledge database to identify the type of fault that occurred. Similarly, the fault compensation component devises a solution to address the fault and executes the recovery process to restore the network to its optimal state.

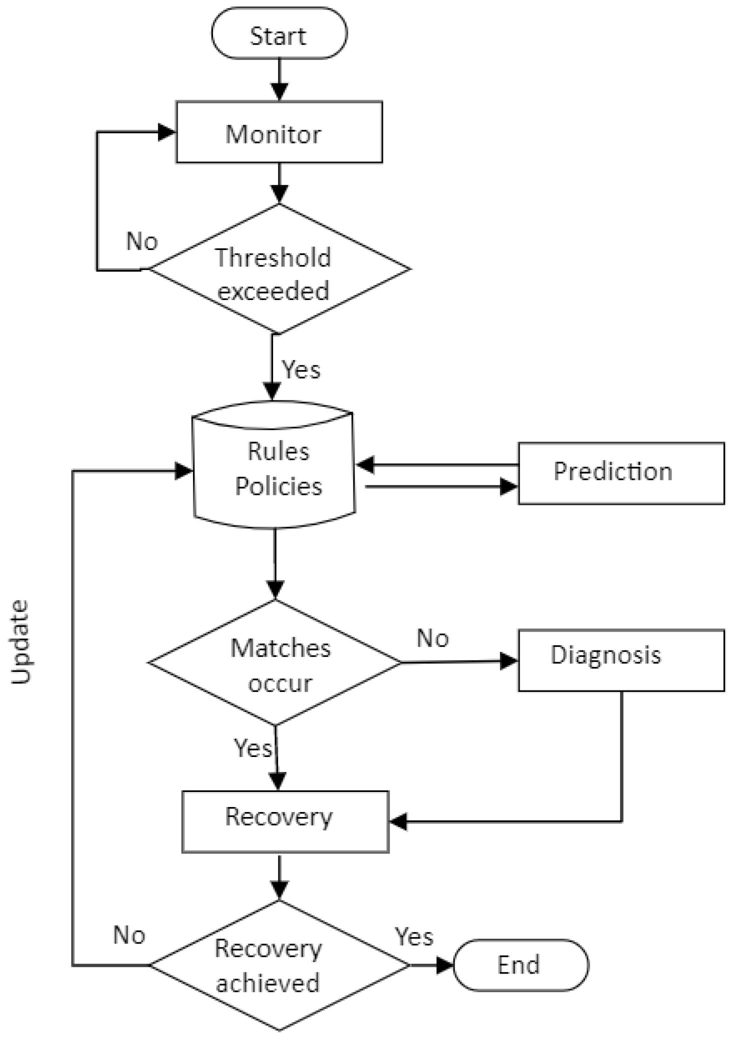


Figure 12. Self-Healing Framework.

Table 7 summarizes the artificial intelligence (AI) methods that have been proposed for self-healing, categorized according to the three major self-healing components described in Figure 12. The table also specifies the algorithms utilized for each approach.

Below are some examples of approaches taken in the literature to achieve self-healing systems:

In [76], a self-healing framework based on a Bayesian network was proposed for self-diagnosis in wireless networks. The authors utilized a Naive Bayes algorithm to improve the accuracy of the diagnosis process.
In [77], a self-healing method was proposed that combines the support vector machine (SVM) algorithm with the fuzzy logic system to diagnose faults and suggest remedies in cloud computing networks.
In [78], a hybrid approach based on artificial neural networks (ANN) and ant colony optimization (ACO) was proposed for fault detection and diagnosis in wireless sensor networks. The ANN was used to model the relationship between faults and their symptoms, while ACO was utilized to find the shortest path for data transmission in the network.
In [43], a self-healing mechanism for software-defined networks (SDN) was proposed that uses a decision tree algorithm for fault detection and diagnosis. The authors also employed a reinforcement learning algorithm to determine the optimal recovery strategy.

These are just a few examples of the diverse range of AI-based approaches proposed in the literature to achieve self-healing in various types of networks.

Table 7 also highlights the algorithm utilized for each approach. Similarly the section below give a detailed account of some approaches in the literature taken to achieve self-healing systems.

Table 7. AI based methods for Self-Healing in MEC.

Paper	Project Description	Algorithm/ Method	SH Component
[79]	Faults root causes identification through unsupervised clustering	Complete correct call (CCC) rate	Detection and diagnosis
[28]	An automated diagnosis platform with diagnosis model adapted to real UMTS network	Bayesian network (BN)	Diagnosis
[31]	Fault diagnosis through a supervised genetic learning algorithm, with vectors of PIs, alarms, and configuration parameters as input	Supervised genetic fuzzy algorithm	Diagnosis
[80]	Management of fault by means of real time information learning and analysis	Bayesian network	Diagnosis, prediction
[81]	Fault management in an outdoor cellular network	Deep reinforcement learning	Utilizes alarms for detection
[82]	Analysis of network logs to find KPI and the respective faulty network element	Time series decomposition	Detection
[5]	Utilizes reference signal received power (RSRP) to identify performance degradation	Random forest and convolutional neural network	Detection, diagnosis

Rezaei et al. [79] proposed a fault diagnosis method for cellular networks using key performance indicators (KPI) from operational support systems data. The approach relied on unsupervised clustering of KPIs, and the authors concluded that fault detection and diagnosis is possible when sufficient historical data on faults and root causes are available. However, considering the multiple KPIs used in cellular networks, each with its own set of measuring standards, harmonizing diverse KPIs can be time-consuming and may lead to thresholds that are either too lenient or too rigorous for their intended purpose. Moreover, the proposed method does not account for situations where sufficient historical data on failures and expert knowledge are unavailable.

Similarly, Khalil et al. [83] presented self-healing neural networks as an approach for automatic fault detection and correction. The authors utilized a shared operation with a spare node at each layer, and the spare node is designed to compensate for any faulty node in the layer. However, the introduction of a spare node at each layer raises concerns about hardware area overhead, which is not entirely eliminated by this approach.

Mismar and Evans [81] proposed a deep Q-learning algorithm as a fault management strategy for an outdoor cellular network. They implemented deep reinforcement learning to enable self-healing of the network by manipulating different alarm corrective actions. The algorithm aims to maximize the downlink signal-to-interference-plus-noise ratio, resulting in improved network performance.

Barco et al. [84] introduced a unified structure for self-healing in a network, which automatically carries out the different functions required for self-healing. The first function is information gathering from various sources, including key performance indicators, mobile traces, alarms, real-time monitoring, configuration parameters, network counters, context information, and drive tests. The second function is fault detection, which focuses on identifying service components and degraded components. The third function is diagnosis, which includes fault identification and rectification. The fourth function is fault compensation, which aims to provide an alternative solution to minimize service degradation while the fault is being fixed. The final function is reporting, which documents the steps taken in solving the problem for future reference.

Gurgen et al. [85] propose the MAPE-K model for self-healing of online sensor data. The framework allows for the implementation of several algorithms that can detect and classify faults based on the properties of the types of sensor data, as well as the sensor network topology.

The authors adapt the MAPE-K concept to create self-healing capabilities for cyber-physical systems in smart buildings and communities. Their approach consists of a four-phase control loop, i.e., monitor, analyze, plan, and execute, which engages continuously with a knowledge base. In the monitoring phase, various characteristics of the components, such as actuators, sensors, and gateways, are continuously monitored. In the analysis phase, the data obtained from the monitoring stage is extracted and analyzed to provide useful information on the status of the components. The planning phase makes decisions on the steps to be taken to accomplish the specific targets set by managers. Prediction models such as Bayesian networks, decision trees, or fuzzy logic can be utilized in this phase. The execution stage plans and conducts the determined actions as decided in the previous phase.

Begum and Nandury [86] proposed a component-based self-healing strategy to address link failures in wireless sensor networks. Specifically, in the case of an intermediate node breakdown, the solution focuses on building an alternative route to the root node. The authors offered two techniques for selecting an alternate path without breaking the precedence relations when there is a defective node. However, the approach is reactive and does not take into account the importance of predicting and preventing node failure. A more proactive approach could be developed to complement the reactive approach and improve the overall fault management of the network.

Gomez et al. [87] focused on the design and evaluation of self-healing in long-term evolution (LTE) networks by developing a system model to generate faults and measure key performance indicators. The paper proposed a methodology to extract specialist knowledge necessary to build a self-healing system from a collection of reported incidents. Finally, a fuzzy-logic-based system was utilized to pinpoint the faults in LTE networks. However, the study's scope was limited to identifying service deterioration, and a more comprehensive approach could be taken to include a proactive method for preventing degradation from occurring in the first place. Nonetheless, this paper provides valuable insights into the design and evaluation of self-healing systems for LTE networks.

Hellbr and Fischer [88] introduced a model-based approach to enable self-healing capabilities in IoT systems. Their strategy allows for easy integration of these capabilities into existing systems. The paper outlines three essential modules- monitoring, diagnosis, and mitigation, which work together in a loop, along with the MAPE-K model to achieve self-healing capabilities.

The monitoring module comprises one or more monitors, each of which is responsible for monitoring one aspect of the system or device. The diagnosis module determines the cause of the failure based on the results from the monitoring module and selects the most effective mitigation action. The mitigation module executes the selected mitigation action to restore the system to its correct state. However, a more proactive approach could have been considered by the authors to prevent failures from happening in the first place.

6.4. Proactive Self-Healing Systems

Self-healing is a key aspect of the autonomic computing paradigm, inspired by the human nervous system's ability to respond to various stimuli. An autonomous entity possesses the capability to make decisions independently, which is the basis of the three main elements of autonomic computing: self-configuration, self-healing, and self-optimization.

Self-healing refers to a network's ability to restore its operations after a failure has occurred, without any external intervention. A self-healing system can detect, diagnose, and provide solutions to failures.

Self-healing can be achieved through either a reactive or proactive approach. In a reactive approach, the system responds to the fault after it has occurred, while a proactive approach seeks to prevent the fault from happening by predicting it beforehand.

Traditional self-healing methods typically wait for a fault to occur before the system takes action to discover, diagnose, and remedy the issue. Proactive self-healing, on the other hand, continually monitors for threats and anticipates failures before they occur, taking steps to prevent them.

Proactive self-healing is a concept that builds on proactive fault tolerance, which aims to predict and solve problems before they occur. Proactive fault tolerance encompasses processes such as failure prediction, prediction accuracy, and solution implementation and has been instrumental in achieving high levels of reliability and service availability.

This technique has been successfully applied in various fields. For example, Ray et al. [89] used proactive fault tolerance to predict the failure of computers in a federation based on CPU temperature. Similarly, Tuli et al. [90] investigated the bottleneck caused by job migration in a network and proposed PreGAN, a system that leverages a generative adversarial network (GAN) to achieve proactive fault tolerance in networked containers.

Proactive fault tolerance has been applied in various fields, including network failures. Park [91] presented a proactive approach to solving network failures, formulating the problem as a constrained optimization problem with traffic demand, links, and the routing layer's condition as key constraints. The network's robustness was considered the objective function. To achieve stability, the authors proposed optimizing routing pathways, link scheduling, and traffic production rate. In another study on proactive fault tolerance for virtual machine failures, Rawat et al. [92] proposed a time series stochastic model. The method involved monitoring virtual machine execution and recording health-related data. The proactive fault tolerance concept is also applicable for predicting security problems that may impact network services and providing appropriate countermeasures. Ge et al. [93] proposed an integrated proactive defense system capable of deceiving attackers by creating a network topology consisting of both decoy and authentic nodes. A genetic algorithm was used to shuffle the nodes, making it difficult for attackers to target the network.

The role of switches and controllers in networking is critical, and it is important to proactively protect the network through these devices. To this end, Padma and Yogesh [94] proposed a proactive strategy that uses rapid recovery techniques within the switch and controller. This approach allows the controller to determine backup pathways using the link protection technique and proactively install them alongside functional paths, while enabling the switches to undertake recovery steps locally. However, this method increases the number of backup path entries, resulting in poor bandwidth efficiency.

In addition, it is worth noting that there are other proactive measures that can be taken to protect network switches and controllers. For example, Zhao et al. [47] proposed a dynamic load balancing approach that utilizes the SDN controller to distribute traffic load across the network. This approach proactively improves network performance by distributing traffic to available resources and avoiding network congestion.

Proactive self-healing systems can be achieved through various methods. Goureb et al. [95] propose a method for load balancing across multiple controllers by assigning switches to controllers and using a min–max model to reduce the overall load of the controllers.

Another example of proactive self-healing is Santos et al.'s [62] SELFNET architecture for 5G networks. The framework consists of several interworking layers that detect and address network failures and malfunctions. Sensors gather information, which is analyzed to determine the appropriate countermeasures. Actuators then enforce the recovery actions to restore the network to an optimal state.

Additionally, Ashgar et al. [96] utilized information from the users environment together with traditional information largely acquired from the network through the use of KPI thresholds to proactively provide self-healing response in a cellular network.

In addition to these methods, researchers have also used various algorithms, such as SVM and neural networks, to predict failures and anticipate problems proactively. However, some of these studies did not use real network data, which could limit their accuracy. In contrast, Kumar et al. [97] applied a Continuous Time Markov Chain (CTMC) to analyze network reliability using actual network data. By using transition probabilities to capture past faults, they were able to formulate a predictive model with the dual capability of CTMC. Their analysis showed a high probability of the network transitioning from a healthy to an unhealthy state, providing valuable information about network reliability.

To enhance the self-healing process in cellular networks, a proactive approach is required. This approach involves adding a prediction component to the traditional detection, diagnosis, and recovery components of self-healing. The primary purpose of the prediction component is to anticipate faults before they occur. Typically, the traditional approach to self-healing in cellular networks involves gathering metrics to evaluate the network's current state. The gathered information is then synthesized into key performance indicators (KPIs). The KPIs can also be created from a network without a known fault, and they are used to build a network profile. The KPIs are continually monitored, and any deviations from the profile will trigger an alarm.

A proactive approach to self-healing requires the addition of a prediction component to the previously mentioned detection, diagnosis, and recovery components of SH. The function of this component is to predict failure before it happens. The prediction component utilizes the past and current information gathered from the network to predict the occurrence of faults that is about to happen. Faults may progressively develop into failure; hence, the prediction component seeks to capture these faults and anomalies early on and pass them to other SH components to figure out what failure it may develop into and how to resolve it. The prediction component achieves its goal by continuously monitoring the network for symptoms and tracking failures to generate enough data, which helps in increasing the accuracy of the prediction [98].

6.5. Discussion

MEC, or multi-access edge computing, is a technology that brings computation, storage, and networking closer to end users, providing ultra-low latency and high bandwidth [99]. This has opened up numerous opportunities for businesses. However, the limited resources at the edge compared to the cloud, as well as the limited resources of mobile end users, have created challenges such as resource allocation and task offloading [100,101]. Additionally, the MEC environment is prone to issues, such as network management, network attacks, and security and privacy concerns [102,103].

This study aims to explore the utilization of self-healing (SH) approaches in mobile edge computing (MEC) environments to address various challenges [104,105]. Additionally, the study seeks to identify suitable design theories, methods, and algorithms that can support SH in MEC. The research results suggest that the MAPE-k model can offer autonomous functionality to an existing MEC network [106,107].

The SH approach in the MEC environment is comprised of several components that work together to monitor, analyze, plan, and execute actions to restore a system to its functional state [108,109]. In addition to this, another notable approach to SH in the MEC environment is the integration of technologies such as SON, SDN, and NFV. By integrating SDN and NFV, SON capabilities are enhanced, making the network more dynamic and programmable and, therefore, more adaptable to self-healing components [110,111].

The Kubernetes approach focuses on the services and applications running on the MEC network [112,113]. It ensures that the containers or microservices running on the platform function continuously without fail. Kubernetes leverages its intrinsic SH properties to ensure that all clusters operate optimally.

The goal of this paper is to explore proactive approaches in network management to prevent system breakdowns, as opposed to relying on reactive approaches [114,115]. Proactive methods are beneficial because they promote continuous service availability, which enhances user experience, profitability, and overall business growth.

This paper will primarily focus on network management theories applicable to MEC, with particular emphasis on autonomous functionalities [116,117]. We will review existing autonomic architectures and frameworks that support autonomous functionalities. It is widely acknowledged that a system's ability to automatically provide solutions when problems arise is of utmost importance.

Small and large organizations alike can benefit from mobile edge computing (MEC), due to its ability to provide significant performance enhancements in several areas such as augmented reality, video analytics, location services, the Internet of Things (IoT), linked cars, optimized local content distribution, and data caching [118,119].

Mobile edge computing is beneficial for organizations of all sizes [120,121]. It offers improved performance in areas such as augmented reality, linked cars, video analytics, location services, IoT, optimised local content distribution, and data caching. Additionally, MEC creates opportunities for new business sectors and services for both consumers and businesses. An excellent example of this is the partnership between Vodafone Business and Amazon Web Services, which leverages MEC to accelerate 5G deployment. Furthermore, by moving processing closer to the data source, organizations can realize the value of data assets in real-time, enabling them to make critical business decisions [122].

MEC is considered an essential enabler of intelligent automation and a key infrastructure component for the full potential realization of emerging technologies, including 5G, IoT, and AI. Although MEC architecture accommodates heterogeneous technologies and complex network infrastructure, managing this dense and complex infrastructure amid possible failure or attack is crucial in light of the increasing data and uncompromisingly high expectations of users in terms of quality of service. Managing such a complex system like MEC requires autonomous functionalities that ensure optimal quality of service (QoS) and quality of experience (QoE). Relying solely on human effort cannot provide an adequate solution. Therefore, self-healing is a crucial concept for network management in the information age.

7. Conclusions

MEC provides a solution to the need for better quality of service by bringing computation closer to the subscribers, but managing both the infrastructure and services can be complex. The autonomous functionalities of MEC and SON, which consist of self-configuration, self-healing, self-optimization, and self-protection, can help to manage the complexity of the system. By incorporating these autonomous features, MEC and SON can perform tasks such as configuration, healing, optimization, and protection without human intervention. This reduces downtime, improves system resilience, and enhances the quality of service, thereby contributing to a better user experience.

This paper has provided a comprehensive review of past research conducted to achieve autonomous behavior in various fields. Additionally, the architecture and framework of autonomous systems were examined. The approaches for achieving self-healing were categorized into model-based, component-based, and the MAPE-K model. Moreover, the utilization of algorithms such as fuzzy logic, Bayesian networks, and decision trees, as part of the self-healing components, was highlighted. The paper concludes that the development of autonomous systems has become necessary, due to the complexities in the management of infrastructure and services, particularly in the mobile edge computing environment. Furthermore, achieving high availability of IT infrastructure is critical for businesses to maintain competitive advantage and continuity, as downtime could result in revenue loss and negative publicity. Thus, the need for an autonomous process that requires no human intervention to handle complexities is evident, with the self-healing, self-optimization, self-protection, and self-configuration features being the core components of autonomous functionalities. Future research could focus on the development of more advanced and efficient autonomous systems that can handle more complex and dynamic systems in various fields.

Finally, in this study, a variety of interesting insights were offered. Firstly, the findings of the study showed that self-healing methodologies were embraced by various IT fields to address the need for autonomous network management. The study also demonstrated that the recent literature had paid attention to the MEC resource management strategy, which includes network infrastructure and task offloading, in regards to self-healing. However, the MEC still faced challenges related to resource allocation, task offloading, and security. The SLR procedure was complex and required reading multiple publications. Electronic data management software was used to gather, organize, and analyze information, which improved the review accuracy and process efficiency.

Using the findings of this SLR as a foundation, future work would focus on the use of proactive SH to address security issues in the MEC environment. Future work could also focus on exploring the integration of autonomous functionalities in MEC and SON to address the challenges related to resource allocation, task offloading, and security in MEC. Specifically, proactive self-healing could be utilized to improve security issues in the MEC environment. Additionally, future research could focus on developing more advanced and efficient autonomous systems that can handle even more complex and dynamic systems in various fields, including MEC. Another avenue for future research could be exploring the use of machine learning and artificial intelligence algorithms in the development of autonomous functionalities in MEC and SON. Finally, the paper highlights the complexity of the SLR procedure and the use of electronic data management software to improve review accuracy and process efficiency, suggesting that future work could investigate other approaches to conducting systematic reviews that can streamline the process, while maintaining quality.

Author Contributions: Conceptualization, O.A.; methodology, O.A.; validation, A.S.S. and P.P.; formal analysis, O.A.; investigation, O.A.; resources, M.A.T.; writing—original draft preparation, O.A.; writing—review and editing, A.S.S. and O.K.; supervision, A.S.S. and P.P.; All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data will be made available to researchers with an individual request to either the first author or the corresponding author, with restrictions that it will be used for literature progress only.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Abbas, N.; Zhang, Y.; Taherkordi, A.; Skeie, T. Mobile Edge Computing: A Survey. *IEEE Internet Things J.* **2017**, *10*, 450–465. [CrossRef]
2. George, G.M.; Jayashree, L.S. Fusion of Blockchain-IoT network to improve supply chain traceability using Ethermint Smart chain: A Review. *KSII Trans. Internet Inf. Syst.* **2022**, *16*, 3694–3722. [CrossRef]
3. El Haber, E.; Nguyen, T.M.; Assi, C.; Ajib, W. Macro-cell assisted task offloading in MEC-based heterogeneous networks with wireless backhaul. *IEEE Trans. Netw. Serv. Manag.* **2019**, *16*, 1754–1767. [CrossRef]
4. Nikmanesh, S.; Akbari, M.; Joda, R. Proactive Self-Healing Analysis-Framework Based on Discrete-Time Markov Decision Process in 5G Network and beyond. In Proceedings of the 9th International Symposium on Telecommunications (IST), Tehran, Iran, 17–19 December 2018; pp. 690–695.
5. Porch, J.B.; Foh, C.H.; Farooq, H.; Imran, A. Machine learning approach for automatic fault detection and diagnosis in cellular networks. In Proceedings of the 2020 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom), Odessa, Ukraine, 26–29 May 2020; pp. 1–5.
6. Kephart, J.O.; Chess, D.M. The vision of autonomic computing. *Computer* **2003**, *36*, 41–50. [CrossRef]
7. Rouse, M. Network Node. 2016. Available online: https://searchnetworking.techtarget.com/definition/node (accessed on 29 February 2021).
8. Jason, H. Architecture of Autonomic Computing Explained. 2020 Available online: https://wisdomplexus.com/blogs/architecture-autonomic-computing/ (accessed on 14 December 2020).
9. Avinetworks. High Availability. 2021. Available online: https://avinetworks.com/glossary/high-availability/ (accessed on 3 March 2021).
10. Network and Servers. Networks and Servers Technologies and Tendencies in Cyberspace. 2019. Available online: https://networksandservers.blogspot.com/2011/02/high-availability-terminology-i.html (accessed on 3 March 2021).
11. Pan, J.; McElhannon, J. Future edge cloud and edge computing for internet of things applications. *IEEE Internet Things J.* **2017**, *5*, 439–449. [CrossRef]
12. Kaur, K.; Dh, T.; Kumar, N.; Zeadally, S. Container-as-a-service at the edge: Trade-off between energy efficiency and service availability at fog nano data centers. *IEEE Wirel. Commun.* **2017**, *24*, 48–56. [CrossRef]
13. Satyanarayanan, M.; Bahl, P.; Caceres, R.; Davies, N. The Case for VM-Based Cloudlets in Mobile Computing. *IEEE Pervasive Comput.* **2009**, *8*, 14–23. [CrossRef]
14. Condoluci, M.; Mahmoodi, T. Softwarization and virtualization in 5G mobile networks: Benefits, trends and challenges. *Comput. Netw.* **2018**, *146*, 65–84. [CrossRef]
15. Fadlullah, Z.M.; Tang, F.; Mao, B.; Kato, N.; Akashi, O.; Inoue, T.; Mizutani, K. State-of-the-art deep learning: Evolving machine intelligence toward tomorrow's intelligent network traffic control systems. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 2432–2455. [CrossRef]
16. Kim, H.G.; Park, S.; Lange, S.; Lee, D.; Heo, D.; Choi, H.; Yoo, J.H.; Hong, J.W.K. Graph neural network-based virtual network function deployment optimization. *Int. J. Netw. Manag.* **2021**, *10*, 2164. [CrossRef]
17. Moher, D.; Liberati, A.; Tetzlaff, J.; Altman, D.G.; PRISMA Group. Preferred reporting items for systematic reviews and meta-analyses: The PRISMA statement. *Ann Intern Med.* **2009**, *151*, 264–269. [CrossRef] [PubMed]
18. Parker, C.; Scott, S.; Geddes, A. Snowball sampling. In *SAGE Research Methods Foundations*; SAGE Publications: Thousand Oaks, CA, USA, 2019.
19. Hsieh, L.T.; Liu, H.; Guo, Y.; Gazda, R. Task Management for Cooperative Mobile Edge Computing. In Proceedings of the 2020 IEEE/ACM Symposium on Edge Computing (SEC), Online, 12–14 November 2020; pp. 352–357.
20. Wu, C.; Peng, Q.; Xia, Y.; Lee, J. Mobility-aware tasks offloading in mobile edge computing environment. In Proceedings of the 2019 Seventh International Symposium on Computing and Networking (CANDAR), Nagasaki, Japan, 26–29 November 2019; pp. 204–210.
21. Zhang, X.; Xia, C.; Ma, T.; Zhang, L.; Jin, Z. Optimizing Energy-Latency Tradeoff for Computation Offloading in SDIN-Enabled MEC-based IIoT. *KSII Trans. Internet Inf. Syst.* **2022**, *16*, 4081–4098. [CrossRef]
22. Wang, H.; Li, Y.; Jin, D.; Han, Z. Attentional Markov model for human mobility prediction. *IEEE J. Sel. Areas Commun.* **2021**, *39*, 2213–2225. [CrossRef]
23. Kan, T.Y.; Chiang, Y.; Wei, H.Y. Task offloading and resource allocation in mobile-edge computing system. In Proceedings of the 2018 27th Wireless and Optical Communication Conference (WOCC), Hualien, Taiwan, 30 April–1 May 2018; pp. 1–4.
24. Zhang, X.; Debroy, S. Adaptive task offloading over wireless in mobile edge computing. In Proceedings of the 4th ACM/IEEE Symposium on Edge Computing, Arlington, VA, USA, 7–9 November 2019; pp. 323–325.
25. Qiao, G.; Leng, S.; Maharjan, S.; Zhang, Y.; Ansari, N. Deep reinforcement learning for cooperative content caching in vehicular edge computing and networks. *IEEE Internet Things J.* **2019**, *7*, 247–257. [CrossRef]
26. Zoha, A.; Imran, A.; Abu-Dayya, A.; Saeed, A. A machine learning framework for detection of sleeping cells in LTE network. In Proceedings of the Machine Learning and Data Analysis Symposium, Leuven, Belgium, 30 October–1 November 2014.
27. Zoha, A.; Saeed, A.; Imran, A.; Imran, M.A.; Abu-Dayya, A. A learning-based approach for autonomous outage detection and coverage optimization. *Trans. Emerg. Telecommun. Technol.* **2016**, *27*, 439–450. [CrossRef]

28. Khanafer, R.M.; Solana, B.; Triola, J.; Barco, R.; Moltsen, L.; Altman, Z.; Lazaro, P. Automated diagnosis for UMTS networks using Bayesian network approach. *IEEE Trans. Veh. Technol.* **2008**, *57*, 2451–2461. [CrossRef]
29. Ktari, S.; Secci, S.; Lavaux, D. Bayesian diagnosis and reliability analysis of Private Mobile Radio networks. *IEEE Symp. Comput. Commun. (ISCC)* **2017**, *10*, 1245–1250.
30. Navin Dhinnesh, A.D.C.; Sabapathi, T.; Sundareswaran, N. An efficient self-healing network through quadratic probing optimization mechanism. *Int. J. Commun. Syst.* **2022**, *35*, 5098.
31. Khatib, E.J.; Barco, R.; Gómez-Andrades, A.; Serrano, I. Diagnosis based on genetic fuzzy algorithms for LTE self-healing. *IEEE Trans. Veh. Technol.* **2015**, *65*, 1639–1651. [CrossRef]
32. Vilchez, J.M.S.; Yahia, I.G.B.; Crespi, N. Self-healing mechanisms for software defined networks. In Proceedings of the 8th International Conference on Autonomous Infrastructure, Management and Security (AIMS), Brno, Czech Republic, 30 June–3 July 2014.
33. Thorat, P.; Raza, S.M.; Nguyen, D.T.; Im, G.; Choo, H.; Kim, D.S. Optimized self-healing framework for software defined networks. In Proceedings of the 9th International Conference on Ubiquitous Information Management and Communication, Bali, Indonesia, 8–10 January 2015; pp. 1–6.
34. Zhong, X.; Wang, X.; Yang, Y.; Qin, Y.; Ma, X.; Yang, T. A parallel optimal task allocation mechanism for large-scale mobile edge computing. *arXiv* **2020**, arXiv:2005.00537.
35. Tong, Z.; Deng, X.; Ye, F.; Basodi, S.; Xiao, X.; Pan, Y. Adaptive computation offloading and resource allocation strategy in a mobile edge computing environment. *Inf. Sci.* **2020**, *537*, 116–131. [CrossRef]
36. Shan, N.; Cui, X.; Gao, Z. "DRL+ FL": An intelligent resource allocation model based on deep reinforcement learning for Mobile Edge Computing. *Comput. Commun.* **2020**, *160*, 14–24. [CrossRef]
37. Yang, J.; Poellabauer, C. SATSS: A Self-Adaptive Task Scheduling Scheme for Mobile Edge Computing. In Proceedings of the International Conference on Computer Communications and Networks (ICCCN 2021), Athens, Greece, 19–22 July 2021; pp. 1–9.
38. Shakarami, A.; Shahidinejad, A.; Ghobaei-Arani, M. An autonomous computation offloading strategy in Mobile Edge Computing: A deep learning-based hybrid approach. *J. Netw. Comput. Appl.* **2021**, *178*, 102974. [CrossRef]
39. Naouri, A.; Wu, H.; Nouri, N.A.; Dhelim, S.; Ning, H. A novel framework for mobile-edge computing by optimizing task offloading. *IEEE Internet Things J.* **2021**, *8*, 13065–13076. [CrossRef]
40. Lu, H.; Gu, C.; Luo, F.; Ding, W.; Liu, X. Optimization of lightweight task offloading strategy for mobile edge computing based on deep reinforcement learning. *Future Gener. Comput. Syst.* **2020**, *102*, 847–861. [CrossRef]
41. Hu, X.; Zhao, Y.; Huang, Y.; Zhu, C.; Yao, J.; Fang, N. Hierarchical Resource Management Framework and Multi-hop Task Scheduling Decision for Resource-Constrained VEC Networks. *KSII Trans. Internet Inf. Syst.* **2022**, *16*, 3638–3657. [CrossRef]
42. Safavat, S.; Sapavath, N.N.; Rawat, D.B. Recent advances in mobile edge computing and content caching. *Digit. Commun. Netw.* **2020**, *6*, 189–194. [CrossRef]
43. Yang, Z.; Liu, Y.; Chen, Y.; Al-Dhahir, N. Cache-aided NOMA mobile edge computing: A reinforcement learning approach. *IEEE Trans. Wirel. Commun.* **2020**, *19*, 6899–6915. [CrossRef]
44. Li, L.; Zhang, H. Delay optimization strategy for service cache and task offloading in three-tier architecture mobile edge computing system. *IEEE Access* **2020**, *8*, 170211–170224. [CrossRef]
45. Yu, S.; Wang, X.; Langar, R. Computation offloading for mobile edge computing: A deep learning approach. In Proceedings of the IEEE 28th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (IEEE PIMRC 2017), Montreal, QC, Canada, 8–13 October 2017; pp. 1–6.
46. Mukherjee, M.; Matam, R.; Mavromoustakis, C.X.; Jiang, H.; Mastorakis, G.; Guo, M. Intelligent edge computing: Security and privacy challenges. *IEEE Commun. Mag.* **2020**, *58*, 26–31. [CrossRef]
47. Zhao, P.; Huang, H.; Zhao, X.; Huang, D. P 3: Privacy-preserving scheme against poisoning attacks in mobile-edge computing. *IEEE Trans. Comput. Soc. Syst.* **2020**, *7*, 818–826. [CrossRef]
48. Hou, Y.; Garg, S.; Hui, L.; Jayakody, D.N.K.; Jin, R.; Hossain, M.S. A data security enhanced access control mechanism in mobile edge computing. *IEEE Access* **2020**, *8*, 136119–136130. [CrossRef]
49. Xiao, L.; Wan, X.; Dai, C.; Du, X.; Chen, X.; Guizani, M. Security in mobile edge caching with reinforcement learning. *IEEE Wirel. Commun.* **2018**, *25*, 116–122. [CrossRef]
50. Zhang, Y.; Di, B.; Wang, P.; Lin, J.; Song, L. HetMEC: Heterogeneous multi-layer mobile edge computing in the 6 G era. *IEEE Trans. Veh. Technol.* **2020**, *69*, 4388–4400. [CrossRef]
51. Salem, M. What Is an "Autonomous System"? 2018. Available online: https://www.udacity.com/blog/2018/09/what-is-an-autonomous-system.html (accessed on 22 February 2021).
52. Lei, L.; Tan, Y.; Zheng, K.; Liu, S.; Zhang, K.; Shen, X. Deep reinforcement learning for autonomous internet of things: Model, applications and challenges. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 1722–1760. [CrossRef]
53. Wei, X.; Zhao, J.; Zhou, L.; Qian, Y. Broad reinforcement learning for supporting fast autonomous IoTBroad reinforcement learning for supporting fast autonomous IoT. *IEEE Internet Things J.* **2020**, *7*, 7010–7020. [CrossRef]
54. Tahir, M.; Ashraf, Q.M.; Dabbagh, M. Towards enabling autonomic computing in IoT ecosystem. In Proceedings of the 2019 IEEE International Conference on Dependable, Autonomic and Secure Computing, International Conference on Pervasive Intelligence and Computing, International Conference on Cloud and Big Data Computing, International Conference on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech), Fukuoka, Japan, 5–8 August 2019; pp. 646–651.

55. Vieira, K.; Koch, F.L.; Sobral, J.B.M.; Westphall, C.B.; de Souza, Leão, J.L. Autonomic intrusion detection and response using big data. *IEEE Syst. J.* **2020**, *14*, 1984–1991. [CrossRef]
56. Kassimi, D.; Kazar, O.; Boussaid, O.; Merizig, A. New approach for intrusion detection in big data as a service in the cloud. *J. Digit. Inf. Manag.* **2018**, *16*, 259. [CrossRef]
57. Mokhtari, A.; Denninnart, C.; Salehi, M.A. Autonomous task dropping mechanism to achieve robustness in heterogeneous computing systems. In Proceedings of the IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), New Orleans, LA, USA, 18–22 May 2020; pp. 17–26.
58. Yousaf, F.Z.; Bredel, M.; Schaller, S.; Schneider, F. NFV and SDN—Key technology enablers for 5G networks. *IEEE J. Sel. Areas Commun.* **2017**, *35*, 2468–2478. [CrossRef]
59. Solis, B. SDN Research Report 2019: Building Trust in Digital Business. 2019. Available online: https://www.vodafone.com/business/news-and-insights/white-paper/SDN-research-report-2019 (accessed on 5 March 2021).
60. Peterson, L.; Anderson, T.; Katti, S.; McKeown, N.; Parulkar, G.; Rexford, J.; Satyanarayanan, M.; Sunay, O.; Vahdat, A. Democratizing the network edge. *ACM SIGCOMM Comput. Commun. Rev.* **2019**, *49*, 31–36. [CrossRef]
61. Van Steen, M.; Tanenbaum, A.S. A brief introduction to distributed systems. *Computing* **2016**, *98*, 967–1009. [CrossRef]
62. Santos, J.P.; Alheiro, R.; Andrade, L.; Valdivieso Caraguay, A.L.; Barona Lopez, L.I.; Sotelo Monge, M.A.; Garcia Villalba, L.J.; Jiang, W.; Schotten, H.; Alcaraz-Calero, J.M.; et al. SELFNET Framework self-healing capabilities for 5G mobile networks. *Trans. Emerg. Telecommun. Technol.* **2016**, *27*, 1225–1232. [CrossRef]
63. Ramiro, J.; Hamied, K. (Eds.) *Self-Organizing Networks: Self-Planning, Self-Optimization and Self-Healing for GSM, UMTS and LTE*; John Wiley & Sons: New York, NY, USA, 2011.
64. Schmelz, L.C.; Van Den Berg, J.L.; Litjens, R.; Zetterberg, K.; Amirijoo, M.; Spaey, K.; Balan, I.; Scully, N.; Stefanski, S. Self-organisation in wireless networks: Use cases and their interrelations. In Proceedings of the Wireless World Research Forum Meeting, Paris, France, 5–7 May 2009.
65. Alias, M.; Saxena, N.; Roy, A. Efficient cell outage detection in 5G HetNets using hidden Markov model. *IEEE Commun. Lett.* **2016**, *20*, 562–565. [CrossRef]
66. Yu, P.; Zhou, F.; Zhang, T.; Li, W.; Feng, L.; Qiu, X. Self-organized cell outage detection architecture and approach for 5G H-CRAN. *Wirel. Commun. Mob. Comput.* **2018**, *2018*, 6201386. [CrossRef]
67. Palacios, D.; Khatib, E.J.; Barco, R. Combination of multiple diagnosis systems in self-healing networks. *Expert Syst. Appl.* **2016**, *64*, 56–68. [CrossRef]
68. Poniszewska-Marańda, A.; Czechowska, E. Kubernetes cluster for automating software production environment. *Sensors* **2021**, *21*, 1910. [CrossRef] [PubMed]
69. Javed, A.; Malhi, A.; Främling, K. Edge computing-based fault-tolerant framework: A case study on vehicular networks. *Int. Wirel. Commun. Mob. Comput. (IWCMC)* **2020**, *10*, 1541–1548.
70. Ju, L.; Singh, P.; Toor, S. Proactive autoscaling for edge computing systems with kubernetes. In Proceedings of the 14th IEEE/ACM International Conference on Utility and Cloud Computing Companion, Leicester, UK, 6–9 December 2021; pp. 1–8.
71. Casalicchio, E.; Perciballi, V. Auto-scaling of containers: The impact of relative and absolute metrics. In Proceedings of the IEEE 2nd International Workshops on Foundations and Applications of Self* Systems (FAS* W), Tucson, AZ, USA, 18–22 September 2020; pp. 207–214.
72. Zhou, F.; Feng, L.; Kadoch, M.; Yu, P.; Li, W.; Wang, Z. Multiagent RL Aided Task Offloading and Resource Management in Wi-Fi 6 and 5G Coexisting Industrial Wireless Environment. *IEEE Trans. Ind. Inform.* **2021**, *18*, 2923–2933. [CrossRef]
73. Heydari, J.; Ganapathy, V.; Shah, M. Dynamic task offloading in multi-agent mobile edge computing networks. *IEEE Glob. Commun. Conf. (GLOBECOM)* **2019**, *10*, 1–6.
74. Alharbi, T.; Portmann, M.; Pakzad, F. The (in) security of topology discovery in software defined networks. In Proceedings of the 2015 IEEE 40th Conference on Local Computer Networks (LCN), Washington, DC, USA, 26–29 October 2015; pp. 502–505.
75. Azzouni, A.; Boutaba, R.; Trang, N.T.M.; Pujolle, G. sOFTDP: Secure and efficient topology discovery protocol for SDN. *arXiv* **2017**, arXiv:1705.04527.
76. Korhonen, T.; Ylianttila, M.; Hämäläinen, T.D. Self-healing in future mobile networks: From centralized performance optimization to decentralized fault management. *IEEE Commun. Mag.* **2013**, *51*, 97–105.
77. Du, J. Toward self-healing in software-defined wireless mesh networks. *IEEE Wirel. Commun.* **2015**, *22*, 128–137.
78. Elrashidi, M. AI-powered self-healing in wireless networks: Challenges and opportunities. *IEEE Wirel. Commun.* **2020**, *27*, 32–39.
79. Rezaei, S.; Radmanesh, H.; Alavizadeh, P.; Nikoofar, H.; Lahouti, F. Automatic fault detection and diagnosis in cellular networks using operations support systems data. *IEEE/IFIP Netw. Oper. Manag. Symp.* **2016**, *10*, 468–473.
80. Piao, S.; Park, J.; Lee, E. Root cause analysis and proactive problem prediction for self-healing. In Proceedings of the 2007 International Conference on Convergence Information Technology (ICCIT 2007), Gyeongju, Republic of Korea, 21–23 November 2007; pp. 2085–2090.
81. Mismar, F.B.; Evans, B.L. Deep Q-learning for self-organizing networks fault management and radio performance improvement. In Proceedings of the 52nd Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, CA, USA, 28–31 October 2018; pp. 1457–1461.

82. Nediyanchath, A.; Singh, C.; Singh, H.J.; Mangla, H.; Mangla, K.; Sakhala, M.K.; Balasubramanian, S.; Pareek, S. Anomaly Detection in Mobile Networks. In Proceedings of the 2020 IEEE Wireless Communications and Networking Conference Workshops (WCNCW), Seoul, Republic of Korea, 25–28 May 2020; pp. 1–5.
83. Khalil, K.; Eldash, O.; Kumar, A.; Bayoumi, M. Self-healing approach for hardware neural network architecture. In Proceedings of the IEEE 62nd International Midwest Symposium on Circuits and Systems (MWSCAS), Dallas, TX, USA, 4–7 August 2019; pp. 622–625.
84. Barco, R.; Lazaro, P.; Munoz, P. A unified framework for self-healing in wireless networks. *IEEE Commun. Mag.* **2012**, *50*, 134–142. [CrossRef]
85. Gurgen, L.; Gunalp, O.; Benazzouz, Y.; Gallissot, M. Self-aware cyber-physical systems and applications in smart buildings and cities. In Proceedings of the 2013 Design, Automation and Test in Europe Conference and Exhibition, Grenoble, France, 18–22 March 2013; pp. 1149–1154.
86. Begum, B.A.; Nandury, S.V. Component-based Self-Healing Algorithm with Dynamic Range Allocation for Fault-Tolerance in WSN. In Proceedings of the 7th International Conference on Computer and Communication Technology, Allahabad, India, 24–26 November 2017; pp. 58–65.
87. Gómez-Andrades, A.; Muñoz, P.; Khatib, E.J.; de-la-Bandera, I.; Serrano, I.; Barco, R. Methodology for the design and evaluation of self-healing LTE networks. *IEEE Trans. Veh. Technol.* **2015**, *65*, 6468–6486. [CrossRef]
88. Hellbr, H.; Fischer, S. A Model-based Approach for Self-healing IoT Systems Position Paper. *Sensornets* **2018**, *10*, 135–140.
89. Ray, B.K.; Saha, A.; Khatua, S.; Roy, S. Proactive fault-tolerance technique to enhance reliability of cloud service in cloud federation environment. *IEEE Trans. Cloud Comput.* **2020**, *10*, 957–971. [CrossRef]
90. Tuli, S.; Casale, G.; Jennings, N.R. Pregan: Preemptive migration prediction network for proactive fault-tolerant edge computing. In Proceedings of the IEEE INFOCOM 2022-IEEE Conference on Computer Communications London, UK, 2–5 May 2022; pp. 670–679.
91. Park, P. Markov chain model of fault-tolerant wireless networked control systems. *Wirel. Netw.* **2019**, *25*, 2291–2303. [CrossRef]
92. Rawat, A.; Sushil, R.; Agarwal, A.; Sikander, A. A new approach for vm failure prediction using stochastic model in cloud. *IETE J. Res.* **2021**, *67*, 165–172. [CrossRef]
93. Ge, M.; Cho, J.H.; Kim, D.; Dixit, G.; Chen, I.R. Proactive Defense for Internet-of-things: Moving Target Defense With Cyberdeception. *ACM Trans. Internet Technol.* **2021**, *22*, 1–31. [CrossRef]
94. Padma, V.; Yogesh, P. Proactive failure recovery in OpenFlow based software defined networks. In Proceedings of the International Conference on Signal Processing, Communication and Networking (ICSCN), Chennai, India, 26–28 March 2015; pp. 1–6.
95. Gouareb, R.; Friderikos, V.; Aghvami, A.H.; Tatipamula, M. Joint reactive and proactive SDN controller assignment for load balancing. *IEEE Globecom Work.* **2019**, *10*, 1–6.
96. Asghar, M.Z., Nieminen, P., Hämäläinen, S., Ristaniemi, T., Imran, M.A. and Hämäläinen, T. Towards proactive context-aware self-healing for 5G networks. *Comput. Netw.* **2017**, *128*, 5–13. [CrossRef]
97. Kumar, Y.; Farooq, H.; Imran, A. Fault prediction and reliability analysis in a real cellular network. In Proceedings of the International Wireless Communications and Mobile Computing Conference (IWCMC), Valencia, Spain, 26–30 June 2017; pp. 1090–1095.
98. Salfner, F.; Lenk, M.; Malek, M. A survey of online failure prediction methods. *ACM Comput. Surv.* **2010**, *42*, 1–42. [CrossRef]
99. Danielle, R. Vodafone Business and Amazon Web Services to Bring Edge Computing Closer to Customers. 2019. Available online: https://www.vodafone.com/business/news-and-insights/company-news/vodafone-business-and-amazon-web-services-to-bring-Edge-computing-closer-to-customer (accessed on 23 November 2021).
100. Zhang, Y.; Yao, J.; Guan, H. Intelligent cloud resource management with deep reinforcement learning. *IEEE Cloud Comput.* **2017**, *10*, 60–69. [CrossRef]
101. Rouse, M. Autonomic Computing. Available online: https://whatis.techtarget.com/definition/autonomiccomputing:~:text=Autonomic%20computing%20is%20a%20self,human%20body's%20autonomic%20nervous%20system.&text=The%20goal%20of%20autonomic%20computing,complexity%20invisible%20to%20the%20user (accessed on 20 December 2020).
102. Zhou, Y.; Cheng, G.; Yu, S. An SDN-enabled Proactive Defense Framework for DDoS Mitigation in IoT Networks. *IEEE Trans. Inf. Forensics Secur.* **2021**, *16*, 5366–5380. [CrossRef]
103. Galuba, W. Outage management through self-healing in distributed mobile networks. In Proceedings of the 5th ACM International Workshop on Mobility Management & Wireless Access, Chania, Greece, 20–22 October 2007.
104. Peng, B.; Seco-Granados, G.; Steinmetz, E.; Fröhle, M.; Wymeersch, H. Decentralized scheduling for cooperative localization with deep reinforcement learning. *IEEE Trans. Veh. Technol.* **2019**, *68*, 4295–4305. [CrossRef]
105. Wang, C.; Wang, J.; Shen, Y.; Zhang, X. Autonomous navigation of UAVs in large-scale complex environments: A deep reinforcement learning approach. *IEEE Trans. Veh. Technol.* **2019**, *68*, 2124–2136. [CrossRef]
106. Zeng, D.; Gu, L.; Pan, S.; Cai, J.; Guo, S. Resource management at the network edge: A deep reinforcement learning approach. *IEEE Netw.* **2019**, *33*, 26–33. [CrossRef]
107. Basir, R.; Qaisar, S.; Ali, M.; Aldwairi, M.; Ashraf, M.I.; Mahmood, A.; Gidlund, M. Fog computing enabling industrial internet of things: State-of-the-art and research challenges. *Sensors* **2019**, *19*, 4807. [CrossRef]
108. Suganuma, T.; Oide, T.; Kitagami, S.; Sugawara, K.; Shiratori, N. Multiagent-based flexible edge computing architecture for IoT. *IEEE Netw.* **2018**, *32*, 16–23. [CrossRef]

109. Jiang, F.; Dong, L.; Wang, K.; Yang, K.; Pan, C. Distributed resource scheduling for large-scale MEC systems: A multi-agent ensemble deep reinforcement learning with imitation acceleration. *IEEE Internet Things J.* **2021**, *9*, 6597–6610. [CrossRef]
110. Peng, H.; Shen, X. Multi-agent reinforcement learning based resource management in MEC-and UAV-assisted vehicular networks. *IEEE J. Sel. Areas Commun.* **2020**, *39*, 131–141. [CrossRef]
111. Leppanen, T. Distributed artificial intelligence with multi-agent systems for MEC. In Proceedings of the International Conference on Computer Communication and Networks (ICCCN), Valencia, Spain, 29 July–1 August 2019; pp. 1-8.
112. Nguyen, D.; Ding, M.; Pathirana, P.; Seneviratne, A.; Li, J.; Poor, V. Cooperative Task Offloading and Block Mining in Blockchain-based Edge Computing with Multi-agent Deep Reinforcement Learning. *IEEE Trans. Mob. Comput.* **2021**, *22*, 2021–2037. [CrossRef]
113. Huang, G.; Youn, H.Y. Proactive eviction of flow entry for SDN based on hidden Markov model. *Front. Comput. Sci.* **2020**, *14*, 1–10. [CrossRef]
114. Fancy, C.; Pushpalatha, M. Proactive Load Balancing Strategy Towards Intelligence-Enabled Software-Defined Network. *Arab. J. Sci. Eng.* **2021**, *10*, 1–8. [CrossRef]
115. Pradeepa, R.; Pushpalatha, M. IPR: Intelligent Proactive Routing model toward DDoS attack handling in SDN. *J. Supercomput.* **2021**, *77*, 12355–12381. [CrossRef]
116. Janabi, A.H.; Kanakis, T.; Johnson, M. Convolutional Neural Network based algorithm for Early Warning Proactive System security in Software Defined Networks. *IEEE Access* **2022**, *10*, 14301–14310. [CrossRef]
117. Kaiwartya, O.; Cao, Y.; Lloret, J.; Kumar, S.; Aslam, N.; Kharel, R.; Abdullah, A.H. and Shah, R.R. Geometry-based localization for GPS outage in vehicular cyber physical systems. *IEEE Trans. Veh. Technol.* **2018**, *67*, 3800–3812. [CrossRef]
118. Nguyen, T.T.; Yeom, Y.J.; Kim, T.; Park, D.H.; Kim, S. Horizontal pod autoscaling in Kubernetes for elastic container orchestration. *Sensors* **2020**, *20*, 4621. [CrossRef]
119. Toka, L.; Dobreff, G.; Fodor, B.; Sonkoly, B. Machine learning-based scaling management for kubernetes edge clusters. *IEEE Trans. Netw. Serv. Manag.* **2021**, *18*, 958–972. [CrossRef]
120. Shah, J.; Dubaria, D. Building modern clouds: Using docker, kubernetes & Google cloud platform. In Proceedings of the IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 6–8 January 2019; pp. 0184–0189.
121. Wang, D.; Zhou, Q.; Partani, S.; Qiu, A.; Schotten, H.D. Mobility prediction Based on Machine Learning Algorithms. In Proceedings of the Mobile Communication-Technologies and Applications—25th ITG-Symposium, Coimbatore, India, 3–4 November 2021; pp. 1–5.
122. Gijsen, B.; Montalto, R.; Panneman, J.; Falconieri, F.; Wiper, P.; Zuraniewski, P. Self-Healing for Cyber-Security. In Proceedings of the 2021 Sixth International Conference on Fog and Mobile Edge Computing (FMEC), Gandia, Spain, 6–9 December 2021; pp. 1–7.