DYNAPARC: AI-Driven Predictive Path Failure Management for Industrial IoT-Fog Networks

Rehab Alawadh
Department of Computer Science
University of York, UK
rehab.alawadh@york.ac.uk

Poonam Yadav
Department of Computer Science
University of York, UK
poonam.yadav@york.ac.uk

Hamed Ahmadi
School of Physics, Engineering and Technology
University of York, UK
hamed.ahmadi@york.ac.uk

***Abstract*—The increasing adoption of IoT-Fog networks in industrial environments demands resilient systems to meet stringent Quality-of-Service (QoS) requirements. Network failures disrupt critical processes and degrade QoS, necessitating innovative predictive failure management. This paper presents the Dynamic Resilient Path Recovery (DYNAPARC) system, an AI-centric solution leveraging Software-Defined Networking (SDN) to predict and mitigate failures in industrial IoT-Fog networks. DYNAPARC integrates AI-based reliability prediction model with SDN's programmable architecture and routing protocols to enhance resilience. A hybrid approach combines proactive and reactive methods: secondary paths are pre-installed (proactively) for immediate failover during primary link failures, while new alternative paths are dynamically calculated in real-time (reactively) for multiple failures, ensuring adaptive routing. To quantify the system's performance, a novel Network Performance Score (N) measures QoS under failure conditions. Simulations show that DYNAPARC maintains an N score above 0.975135 before and after failures, outperforming traditional reactive and proactive methods. Integrating machine learning in the SDN controller significantly reduces packet loss by selecting the most reliable paths. These results highlight the potential of AI-driven prediction and SDN to achieve predictive reliability, ensuring superior resilience, fast recovery, and efficient traffic management in fog-based IIoT environments.**

***Index Terms*—Machine learning (ML), software-defined networks (SDNs), fog computing, industrial Internet of Things (IIoT), resilience, link failure, and routing protocols.**

I. Introduction

The Industrial Internet of Things (IIoT) is poised to become a crucial component of future industrial systems, connecting machines, sensors, and actuators in critical manufacturing environments to enhance efficiency and performance. The integration of fog computing into IIoT can significantly decrease decision-making delay, optimise bandwidth usage, and improve privacy protection [1]. Reliable connections in IIoT require effective communication protocols to ensure low latency, high throughput, minimal packet loss, and efficient data transmission. Enhancing IoT application design is essential for managing the entire infrastructure. Therefore, SDN is employed to provide efficient and reliable IoT management [2]. SDN addresses traditional network limitations by enabling centralised control and programmatic management. It separates routing from traffic forwarding, offering a global view for improved network management [3]. The SDN controller is an essential element tasked with managing network traffic, rerouting it in real-time during link failures, and updating forwarding rules to minimise downtime. While SDN enhances network management, challenges such as link failure handling remain. Industrial IoT applications demand strict control over delay and packet loss due to the dynamic nature of IoT environments, which require near 100% data transmission success and minimal latency [4]. Consequently, SDN must incorporate fault tolerance capabilities to address such issues. Network failures are caused by errors, and management involves detection and recovery methods [5]. In SDN, detection methods are periodic or event-based, while recovery strategies are proactive or reactive, each with its own implementation mechanism [6]. Reactive flow installation is essential after a failure to redirect traffic, but it introduces significant delays due to switch-controller communication issues. The proactive strategy pre-installs flow rules, reducing latency and enabling faster link failure recovery. However, it lacks flexibility for dynamic network changes, leading to potential performance issues and increased storage overhead. Another significant challenge lies in effectively QoS by selecting alternative paths that prioritise reliability while minimising delay. To address the need for pre-failure prediction and post-failure management, this study proposes the Dynamic Resilient Path Recovery (DYNAPARC) framework. The framework dynamically adjusts optimal paths in response to link failures and evolving network conditions. By integrating continuous monitoring protocols and using machine learning to provide the controller with predicted link reliability, the system effectively minimises the computation time required to determine optimal paths and reduces the amount of communication required between the controller and switches. We highlight the major contributions of this paper as follows:

1) Development of a hybrid traffic management strategy combining proactive secondary path pre-installation with reactive path calculation to balance

quick failover and adaptive routing, enhancing resilience to network failures.
2) Introduction of a novel network performance metric (N) to evaluate the hybrid strategy's effectiveness against reactive and proactive approaches, considering QoS parameters like throughput, data transfer, and delay.
3) QoS-optimised path selection leverages machine learning to predict link reliability for critical data transmissions, enhancing network performance.
4) Extensive simulation using TensorFlow, Mininet, and the Ryu controller to evaluate the hybrid strategy against reactive and proactive approaches, followed by assessment before and after integrating AI, highlighting its effectiveness in fog-based IIoT environments.

The structure of this paper is as follows: Section II provides a comprehensive review of relevant studies in the existing literature. Section III outlines the problem statement. Section IV describes the DYNAPARC system model and notation. Section V details the implementation and evaluates the system's performance. Finally, Section VI concludes with future research directions.

II. Related Work

Our previous work [7], introduced the Hybrid Intelligent Fast Failure Recovery (HIFFR) framework, leveraging SDN for network resilience against link failures. It included a pre-failure stage using machine learning for prediction and prevention and a post-failure stage for swift routing adjustments to ensure continuity. While the prior study focused on the framework and theory, this study implements and evaluates it, showcasing its effectiveness in enhancing network resilience through predictive and reactive strategies. A number of studies have examined the management of link failures. Xia et al. [8] present ShareBackup, an architecture that enhances failure recovery in data centres by using a small pool of backup devices to quickly replace failed switches, reducing bandwidth loss. However, it doesn't fully address integration challenges during dynamic changes. The authors in [9], [10] explore how SDN and NFV enhance traffic management. The first study improves load distribution by dynamically adding secondary vSDN controllers, while the second suggests integrating SDN and NFV features for a more agile, efficient, and high-performance network. Their findings highlight the potential of these technologies to optimise network performance. Yang et al. [11] proposed a hybrid IP/SDN strategy using tunnelling and a selection algorithm to reduce delays and optimise bandwidth. However, it lacks consideration of multiple link failures, highlighting the need for further investigation. Isyaku et al. [12] propose an algorithm for backup path computation in SDN, considering flow variabilities, losses, and flow table utilisation. However, it does not evaluate performance under dynamic conditions like topology changes, link failures, and traffic fluctuations. Zheng et al. [13] introduced Sentinel, a framework for SD-WANs that improves recovery efficiency by using backup tunnels to redirect traffic during link failures. It minimises traffic loss and maximises capacity but requires further research on dynamic changes and handling multi-link failures. Petale et al. [14] propose a Group Table-based Rerouting (GTR) technique using OpenFlow's Fast Fail-over feature to optimise memory and reduce resource consumption. Sharma et al. [15] propose a swift recovery approach for OpenFlow networks utilising conventional routing protocols such as BGP and OSPF to manage failures. However, it overlooks implementation complexities and scaling challenges, potentially limiting its practical applicability. Previous studies indicate that current SDN environments often rely on static failover mechanisms that redirect traffic to predefined paths during link failures, but this approach has significant limitations:

1) Pre-computed failover paths do not adapt to real-time network conditions, leading to suboptimal routing decisions.
2) The delay in detecting link failures and implementing rerouting can result in significant downtime and packet loss.
3) The SDN-based literature reroutes data flows along an alternative shortest path by either a reactive or proactive flow installation method.

The proposed DYNAPARC system mitigates the identified limitations by addressing the challenges of previous approaches.

1) DYNAPARC utilises various routing protocols to dynamically calculate optimal paths according to real-time network conditions.
2) Continuous network monitoring enables real-time detection and swift response to link failures, minimising latency and packet loss.
3) QoS considerations include evaluating path reliability levels using machine learning algorithms when computing alternative paths.

By integrating these advanced capabilities, DYNAPARC enhances network resilience, ensuring continuous and reliable service even during and after disruptions.

III. Problem Statement

In modern manufacturing plants, IoT sensors monitor machines for operational status and performance. The data collected from these sensors is critical for maintaining smooth operations and performing predictive maintenance. To handle the vast amount of data generated, fog computing is introduced as an intermediary layer between the IoT devices and the cloud, providing localised data processing and decision-making capabilities as shown in Fig. 1.

This reduces latency and improves real-time responsiveness, which is crucial for time-sensitive applications

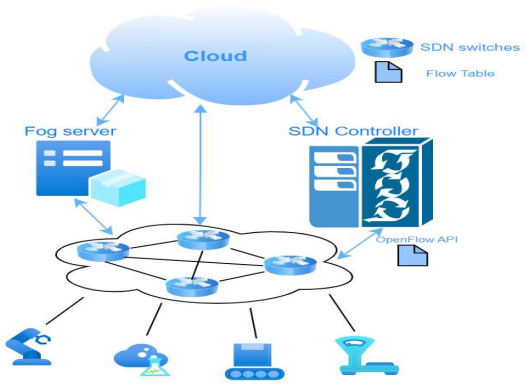


Fig. 1: Architecture of a Software-Defined IIoT System Utilising Fog Computing in a Smart Factory.

in industrial environments. For the experiments, a single fog server is used as a central control point for data aggregation, processing, and decision-making. It receives data from multiple base stations (IoT gateway nodes) in various cities, which collect and transmit sensor data for analysis and action. Reliable traffic characteristics are essential, with key factors including data integrity and guaranteed delivery, ensuring accurate, complete data transmission without loss, and preventing gaps in monitoring. To ensure QoS efficient delivery, three SDN-based strategies were created. The first strategy is reactive, where each SDN switch between the plant and the fog server requests the path from the controller to forward each packet. The second strategy is proactive, where two paths between the plant and the fog server are pre-calculated and stored on the switches. The third strategy is hybrid, which combines both approaches: paths are stored proactively, but in case of link failure, switches request new paths from the controller and store them for future communication.

IV. System Model And Notations

This section provides an overview of the system model used in the DYNAPARC framework, outlining key components and mechanisms that form the basis for the performance evaluation and analysis in the following sections.

A. DYNAPARC System Model

To improve SDN-IIoT network performance, the controller is enhanced by incorporating routing protocols and applications that manage real-time packet rerouting during link failures, as shown in Fig. 2.

The key component of the DYNAPARC system is the SDN controller, which manages network operations, makes routing decisions, and communicates with network devices. The framework comprises five modules that collaborate with the controller to enhance detection and recovery processes. The first module is the **Monitoring Module**, which continuously defines the network structure and provides information about sensor nodes and

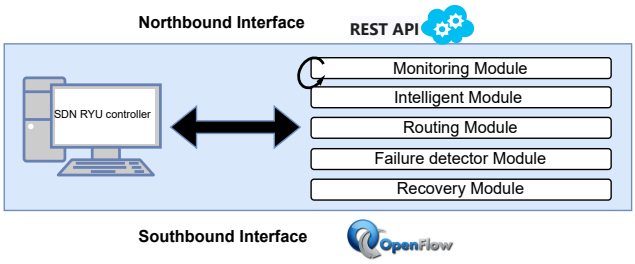


Fig. 2: DYNAPARC System Model.

links. This data is utilised by NetworkX to construct a graph, which is then stored in the controller's memory to streamline the routing process. The **Intelligent Module** analyses traffic flow characteristics to prioritise and route data effectively, using a feedforward neural network machine learning algorithm to route reliable traffic along paths optimised for maximum reliability. The **Routing Module** dynamically computes optimal paths based on real-time network conditions and insights from the Intelligent Module, sharing routing information with all routers and calculating alternative paths even in the absence of link failures. The **Failure Detector Module** constantly observes the network for link faults, quickly identifying communication issues between network entities. This ensures efficient failure detection with rapid response time and minimal overhead. Upon detecting a failure, it notifies the SDN controller, triggering the **Recovery Module**. The **Recovery Module** activates upon failure detection, implementing the pre-computed optimal path to minimise disruption and ensure the continuity of network services. These modules work together to preserve network connectivity in the industrial IoT system during link failures.

B. DYNAPARC system Architecture

The physical structure of the SDN-enabled IIoT-fog system is composed of a collection of SDN switches S, a fog server N, a set of IoT devices D, and a collection of links E, where $E = \{e_{ij} : s_i \leftrightarrow s_j\}$ are the links between nodes represent the communication paths. This system is modelled as a directed graph $H = (S \cup N, E)$. The SDN controller periodically gathers the QoS-related data from the underlying network with intervals of t seconds to optimise the routing strategy for new traffic flows. Each $e_j \in E$ within the network has a set of functions associated with it. The function $B(e_j) : e_j \rightarrow (0, 1]$ is utilised to calculate the throughput of a given link $e_j \in E$, while the function $T(e_j) : e_j \rightarrow (0, 1]$ computes the data transfer for the link. Additionally, the function $D(e_j) : E \rightarrow d_{e_j} > 0$ is employed to determine the transmission delay across the link.

C. network performance score

To evaluate the DYNAPARC system against reactive and proactive, we have developed a Network Performance

Score (N) to assess QoS in identifying the optimal path in the event of a link failure. To define N, we incorporate the target bandwidth, actual data transfer, actual throughput, and delay on the end-to-end path.

$$N = \alpha \left(\frac{B_{\text{actual}}}{B_{\text{target}}} \right) + \beta \left(\frac{T}{T_{\text{max}}} \right) - \gamma D \qquad (1)$$

The variables defined for this analysis are as follows: N represents Network Performance Score, while B_{actual} signifies the actual throughput, measured in kbps. The B_{target} denotes the desired bandwidth allocation, which defines the optimal or expected bandwidth between nodes expressed in kbps. The variable T refers to the total amount of data transferred during a given time window t , quantified in KBytes, and T_{max} indicates the maximum achievable transfer within the network. If the target bandwidth was fully utilised for the duration of the test and is calculated as:

$$T_{\text{max}} = \frac{B_{\text{target}} \cdot t}{8} \quad \text{where} \quad t = 10 \text{ sec} \qquad (2)$$

The variable t indicates the test time interval.
Additionally, D represents time taken for data to travel from one node to another, expressed in ms. The parameters α, β, and γ are defined as weighting factors used in the analysis. The average delay has a negative impact on the N score, and so the N captures the impact of delay on network responsiveness and overall performance, making the metric more adaptable and reflective of real-world conditions. The weighting factors α and β emphasise bandwidth utilisation and data transfer more heavily based on the specific context or requirements. We will consider both $\alpha, \beta = 0.5$ and $\gamma = 0.01$.

V. Implementation And Evaluation Performances

A. Implementation

We evaluate the performance of the proposed approach using the Mininet network emulator in conjunction with the Ryu SDN controller. We created a hypothetical HiberniaUK topology to simulate the network architecture for this case study. This topology is based on the well-known TopologyZoo dataset, which provides realistic network topologies based on actual geographic locations [16]. The HiberniaUK topology consists of 13 nodes representing base stations deployed in different cities across the UK, as shown in Fig. 3.

The design phase involves limiting nodes to three links to mimic real-world scenarios and maintain redundancy. Base stations are connected to nearby neighbours based on geographical proximity to reduce congestion, latency, and high bandwidth costs, creating a cost-effective IIoT network for efficient infrastructure deployment. The experiments are conducted with the objective of comprehensively evaluating the network performance score in the three different controllers: reactive, proactive, and hybrid. In each approach, traffic is systematically generated between the server host *h1* in London and the client host *h6* in Leicester to simulate real-world scenarios. The study analyses the network's response to simulated link failures between switch pairs S1-S5 and S5-S6, assessing the effectiveness and robustness of each approach in managing disruptions and optimising performance.

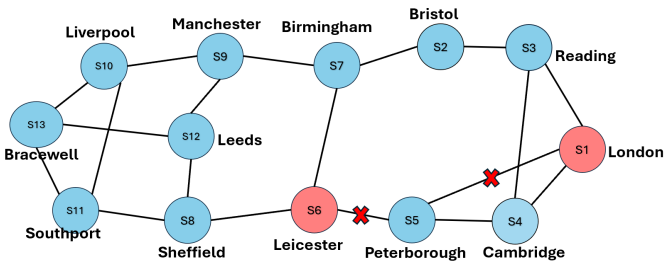


Fig. 3: Network Topology

B. Post-failure management scenario

To evaluate the efficiency of the DYNAPARC system in managing networks after link failures, we conduct an experiment assessing the network's performance by sending packets from host *h1* to host *h6*, allowing the controller to dynamically select the most optimal path as shown in Fig. 3.

Initially, the controller identifies two potential paths: *s1* → *s5* → *s6* as the primary link and *s1* → *s4* → *s5* → *s6* as the secondary link, and selects the most shortest reliable option as shown in Fig. 4. To test the controller's fast recovery capability, we simulate a link failure by dropping the link between switches *s1* and *s5*. The controller then recalculates the routes and identifies the primary alternative paths as *s1* → *s4* → *s5* → *s6* and the secondary path as *s1* → *s3* → *s2* → *s7* → *s6* as shown in Fig. 5. After dropping another link between *s5* and *s6*. The controller again adapts, finding the new paths as the primary link *s1* → *s3* → *s2* → *s7* → *s6* then *s1* → *s4* → *s3* → *s2* → *s7* → *s6* as a secondary path as shown in Fig. 6.

```
------path-1: ['40:57:00:00:00:01', 1, 5, 6, '40:57:00:00:00:06'] ------
------path-2: ['40:57:00:00:00:01', 1, 4, 5, 6, '40:57:00:00:00:06'] ------
```

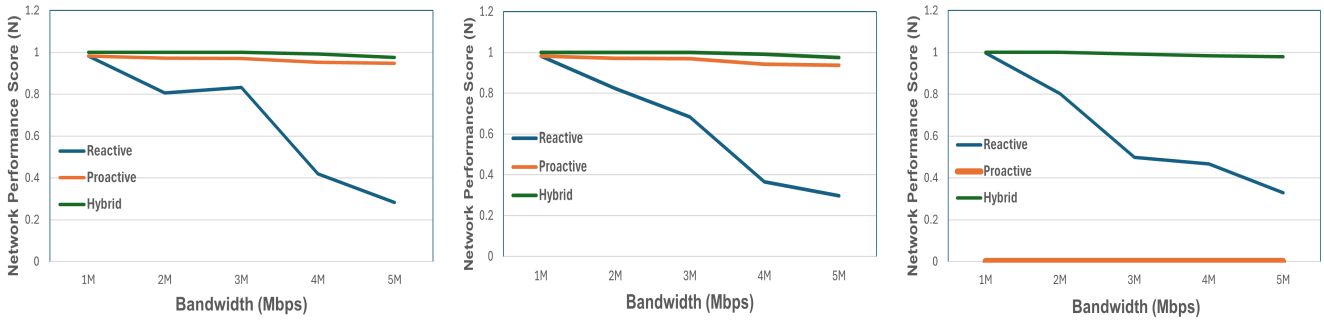
Fig. 4: Initial paths selected.

```
------path-1: ['40:57:00:00:00:01', 1, 4, 5, 6, '40:57:00:00:00:06'] ------
------path-2: ['40:57:00:00:00:01', 1, 3, 2, 7, 6, '40:57:00:00:00:06'] ------
```

Fig. 5: After the first link failure.

```
------path-1: ['40:57:00:00:00:01', 1, 3, 2, 7, 6, '40:57:00:00:00:06'] ------
------path-2: ['40:57:00:00:00:01', 1, 4, 3, 2, 7, 6, '40:57:00:00:00:06'] ----
```

Fig. 6: After the second link failure.



(a) Initial performance. (b) Primary link failure performance. (c) Secondary link failure performance.

Fig. 7: Performance evaluation of different controllers under various conditions.

The experiment demonstrates the DYNAPARC controller's ability to swiftly recover from link failures by selecting optimal alternative paths, minimising disruption, and maintaining high network performance.

C. Performance Evaluation of Approaches

The performance scores N for each strategy, assessed under varying bandwidth loads and different link failure conditions (initial state, primary link failure, and secondary link failure), were calculated using the previously explained formula. These performance scores were evaluated to assess the impact of network load on the effectiveness of each strategy in maintaining QoS and ensuring fast failure coverage, as shown in the Fig. 7. In the **reactive approach,** the performance score decreases as bandwidth increases, dropping from 0.98199 at 1 Mbps to 0.297225 at 5 Mbps, indicating poor performance under higher loads. This decline is observed during both primary and secondary link failures, emphasising the inefficiency of the reactive approach. The **proactive approach,** on the other hand, maintains higher and more consistent performance scores above 0.93663 across all bandwidths. While it shows the same performance during primary link failures, it completely fails to handle secondary link failures, resulting in a score of 0.0. The **hybrid approach,** demonstrates strong performance, with scores consistently exceeding 0.975135 across all bandwidths, similar to the proactive approach. It maintains high performance scores during both primary and secondary link failures, showing its superior reliability compared to the proactive approach, which only performs well during primary link failures. Regarding the impact of network load, The reactive approach experiences significant performance degradation under heavy network load due to its on-demand nature. In contrast, the proactive and hybrid approaches maintain higher performance by pre-installing flows, especially during primary link failures. While the proactive approach ensures quick failover, it fails during secondary link failures. The hybrid approach, however, combines both strategies, offering superior performance and reliability during complex failure scenarios.

D. Pre-failure prediction scenario

In this section, we present a ML-based approach to estimate link reliability in fog-based IIoT environments, ensuring that critical communications—such as automated control signals, safety monitoring systems, and sensor data—are prioritised and operate reliably without delay. The SDN controller continuously monitors the health of the Hibernia UK Network by measuring the utilisation of each link's capacity used to transmit data between plants. These utilisation values are normalised and forwarded to an AI module, which predicts links at risk of failure based on historical data and current usage patterns. To achieve this, we employ a feedforward neural network (FNN) algorithm trained on a real-world network dataset. The FNN model analyses key link parameters, including utilisation rate, repair time, and failure frequency. The AI module plays a critical role by identifying potential failures or congestion points. This allows the SDN controller to dynamically manage traffic and prevent congestion. Such real-time decision-making improves the network's ability to handle high-priority traffic and reduces packet loss by avoiding overloading critical links. In the simulation results, we compare packet loss percentages before and after the integration of AI. The results show the packet loss experienced by three manufacturing plants in the Hibernia UK Network (h7, h11, and h13), which transmit data to the fog server in London (h1). The QoS performance is enhanced by the integration of an AI module, as shown in Fig. 8. Before applying QoS, the packet loss rates were notably high: h7 experienced 30% packet loss, h11 had a significant 60%, and h13 recorded the highest at 70%. Such packet loss rates are unacceptable for critical communications in an IIoT environment, where reliable delivery of data from machines, sensors, and controllers to the fog server is essential. High packet loss can lead to delays, inaccurate readings, or even failures in safety-critical systems, resulting in potential hazards or

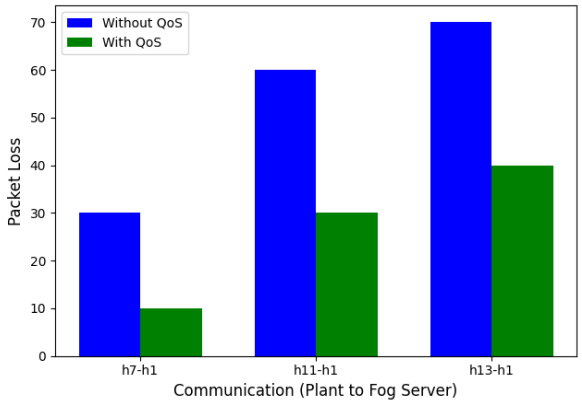


Fig. 8: Packet loss percentages before and after integrating the AI module.

production delays. After applying QoS, enhanced by the integration of AI modules, significant improvements were observed. h7's packet loss dropped to 10%, representing a 67% improvement. Similarly, h11's packet loss decreased to 30%, showing a 50% improvement, while h13's packet loss reduced to 40%, reflecting a 43% improvement. These reductions demonstrate a marked increase in network reliability and data delivery. Combining real-time monitoring and AI, the Hibernia UK IIoT network ensures resilience under high demand and hardware issues.

VI. Conclusion and Future Work

In this study, we introduced the Dynamic Resilience Path Recovery (DYNAPARC) framework, which dynamically adjusts the optimal route based on network link failures and changing conditions. By utilising real-time monitoring, it minimises the path computation time and reduces communication overhead between the SDN controller and switches. The framework integrates multiple network protocols along with an ML algorithm to enhance the performance. This approach enables the dynamic computation of recovery paths, ensuring low latency and reliable packet delivery. DYNAPARC effectively addresses both reactive and proactive network management in SDN-based IIoT networks. For future work, we aim to extend this solution by exploring relevant case studies, focusing on integrating SDN with graph neural networks (GNNs) to predict failures, particularly in real-time networks. This combination of rapid recovery methods and intelligent decision-making will significantly enhance the resilience of the network.

ACKNOWLEDGMENT

Alawadh is funded by the Saudi Arabian Cultural Bureau and Qassim University in Saudi Arabia. Dr. Yadav is supported, in part, by EPSRC and DSIT-funded projects (EP/X040518/l), (EP/Y037421/l), and (EP/Y019229/1).

References

[1] Tie Qiu, Jiancheng Chi, Xiaobo Zhou, Zhaolong Ning, Mohammed Atiquzzaman, and Dapeng Oliver Wu. Edge computing in industrial internet of things: Architecture, advances and challenges. *IEEE Communications Surveys Tutorials*, 22(4):2462–2488, 2020.

[2] Evangelos Haleplidis, Kostas Pentikousis, Spyros Denazis, Jamal Hadi Salim, David Meyer, and Odysseas Koufopavlou. Software-Defined Networking (SDN): Layers and Architecture Terminology. RFC 7426, January 2015.

[3] Adriana Collaguazo Jaramillo, Ronny Alcivar, Joffre Pesantez, and Ronald Ponguillo. Cost effective test-bed for comparison of sdn network and traditional network. In *2018 IEEE 37th International Performance Computing and Communications Conference (IPCCC)*, pages 1–2, 2018.

[4] Abhishek Hazra, Mainak Adhikari, Tarachand Amgoth, and Satish Narayana Srirama. A comprehensive survey on interoperability for iiot: Taxonomy, standards, and future directions. *ACM Comput. Surv.*, 55(1), November 2021.

[5] Paulo César Fonseca and Edjard Souza Mota. A survey on fault management in software-defined networks. *IEEE Communications Surveys Tutorials*, 19(4):2284–2321, 2017.

[6] Babangida Isyaku, Kamalrulnizam Bin Abu Bakar, Fuad A. Ghaleb, and Abdulaziz Al-Nahari. Dynamic routing and failure recovery approaches for efficient resource utilization in openflow-sdn: A survey. *IEEE Access*, 10:121791–121815, 2022.

[7] Rehab Alawadh, Poonam Yadav, and Hamed Ahmadi. Hiffr: Hybrid intelligent fast failure recovery framework for enhanced resilience in software defined networks. In *2024 11th International Conference on Wireless Networks and Mobile Communications (WINCOM)*, pages 1–7, 2024.

[8] Yiting Xia, Xin Sunny Huang, and T. S. Eugene Ng. Stop rerouting! enabling sharebackup for failure recovery in data center networks. In *Proceedings of the 16th ACM Workshop on Hot Topics in Networks*, HotNets '17, page 171–177, New York, NY, USA, 2017. Association for Computing Machinery.

[9] Sikandar Ejaz, Zeshan Iqbal, Peer Azmat Shah, Bilal Haider Bukhari, Armughan Ali, and Farhan Aadil. Traffic load balancing using software defined networking (sdn) controller as virtualized network function. *IEEE Access*, 7:46646–46658, 2019.

[10] Timothy Wood, K. K. Ramakrishnan, Jinho Hwang, Grace Liu, and Wei Zhang. Toward a software-based network: integrating software defined networking and network function virtualization. *IEEE Network*, 29(3):36–41, 2015.

[11] Ze Yang and Kwan L. Yeung. Sdn candidate selection in hybrid ip/sdn networks for single link failure protection. *IEEE/ACM Trans. Netw.*, 28(1):312–321, February 2020.

[12] Babangida Isyaku, Kamalrulnizam bin Abu Bakar, Muhammad Nura Yusuf, and Mohd Soneri Mohd Zahid. Software defined networking failure recovery with flow table aware and flows classification. In *2021 IEEE 11th IEEE Symposium on Computer Applications Industrial Electronics (ISCAIE)*, pages 337–342, 2021.

[13] Lilei Zheng, Hongli Xu, Suo Chen, and Liusheng Huang. Performance guaranteed single link failure recovery in sdn overlay networks. In *2020 IEEE 26th International Conference on Parallel and Distributed Systems (ICPADS)*, pages 703–708, 2020.

[14] Shrinivas Petale and Jaisingh Thangaraj. Link failure recovery mechanism in software defined networks. *IEEE Journal on Selected Areas in Communications*, 38(7):1285–1292, 2020.

[15] Sachin Sharma, Didier Colle, and Mario Pickavet. Enabling fast failure recovery in openflow networks using routeflow. In *2020 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN*, pages 1–6, 2020.

[16] N. Falkner R. Bowden S. Knight, H. Nguyen and M. Roughan. The internet topology zoo, 2024. Accessed: 2024-05-29.