Investigating the dependability of software-defined IIoT-Edge networks for next-generation offshore wind farms

Agrippina Mwangi, Nadine Kabbara, Patrick Coudray, Mikkel Gryning, Madeleine Gibescu

***Abstract*—Next-generation offshore wind farms are increasingly adopting vendor-agnostic software-defined networking (SDN) to oversee their Industrial Internet of Things Edge (IIoT-Edge) networks. The SDN-enabled IIoT-Edge networks present a promising solution for high availability and consistent performance-demanding environments such as offshore wind farm critical infrastructure monitoring, operation, and maintenance. Inevitably, these networks encounter stochastic failures such as random component malfunctions, software malfunctions, CPU overconsumption, and memory leakages. These stochastic failures result in intermittent network service interruptions, disrupting the real-time exchange of critical, latency-sensitive data essential for offshore wind farm operations. Given the criticality of data transfer in offshore wind farms, this paper investigates the dependability of the SDN-enabled IIoT-Edge networks amid the highlighted stochastic failures using a two-pronged approach to: (i) observe the transient behavior using a proof-of-concept simulation testbed and (ii) quantitatively assess the steady-state behavior using a probabilistic Homogeneous Continuous Time Markov Model (HCTMM) under varying failure and repair conditions. The study finds that network throughput decreases during failures in the transient behavior analysis. After quantitatively analyzing 15 case scenarios with varying failure and repair combinations, steady-state availability ranged from 93% to 98%, nearing the industry-standard SLA of 99.999%, guaranteeing up to 3 years of uninterrupted network service.**

***Index Terms*—Industrial IoT, software-defined networking, edge computing, IEEE802.1 Time Sensitive Networking, IEC61850, vPAC, Homogeneous CTMM, offshore wind, dependability**

I. Introduction

A. Problem Definition and Related Works

The increasing complexity of offshore wind turbine and wind farm designs, combined with the intricacies of the marine environment warrants the continued deployment of a wide array of industry-grade Internet of Things (IIoT) sensors for critical infrastructure monitoring. These IIoT sensors monitor and communicate the status of various critical mechanical and electrical components [1], [2]. Further, these IIoT sensors communicate with each other to exchange environmental data and coordinate actions among the turbines in the wind farm to prevent damage or accidents. This reduces the risk of catastrophic failure and subsequent unplanned wind farm downtime [3]. Notably, the unreliability of critical infrastructure in wind farms results in $\geq$87% of the total unplanned wind farm downtime [4]. This unreliability impacts the wind farm's revenue by a 20% rise in operation and maintenance (O&M) costs [5], [6], which constitute 10-20% of the total wind farm's energy costs. Mounting IIoT sensors on key wind farm components help the O&M personnel make prompt decisions facilitating condition-based maintenance subsequently reducing overall wind farm downtime [1], [7] and associated operational expenses.

Contemporary IIoT platforms founded on the IoT/cloud framework encounter high latency, substantial data transfer, storage subscription expenses, intricate scaling processes, bandwidth limitations, and restricted connectivity [8]. Therefore, incorporating edge computing to allow the processing of IIoT sensor data samples closer to the wind farm turbines is a favored solution. Empirical studies and probabilistic models [9]–[13] demonstrate improved wind farm monitoring, maintenance, and operations when data is processed close to offshore wind farms through reduced latency, bandwidth optimization, and advanced computing and storage.

In this IIoT-Edge computing paradigm, next-generation offshore wind farms are increasingly adopting vendor-agnostic software-defined networking (SDN) solutions for dynamic and centralized network management and configuration of their IIoT-Edge networks. In principle, SDN separates the control plane from the data plane [14]. The control plane comprises several controllers that make the forwarding decisions, package them as flow instructions, and send them to the data plane through a southbound interface protocol such as OpenFlow [15], [16]. The data plane forwarding devices (FDs) receive and store the flows in their respective flow tables instructing them on how to forward the data [14], [17], [18]. Unfortunately, the proposed SDN-enabled IIoT-Edge network is susceptible to stochastic failures such as random component malfunction, software malfunction, CPU overconsumption, and memory leakages which cause intermittent network service disruption. This affects the real-time exchange

This work is part of the Innovative Tools for Cyber-Physical Energy Systems (InnoCyPES) project that has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska Curie grant agreement No. 956433.

Agrippina Mwangi and Madeleine Gibescu are with the Energy and Resources Group of the Copernicus Institute of Sustainable Development at Utrecht University, The Netherlands. (Corresponding author email: a.w.mwangi@uu.nl)

Nadine Kabbara and Patrick Coudray are with the Système Départment at EDF R&D, Paris Saclay, France.

Mikkel Gryning is with the SCADA and communications, EPCO at Ørsted Wind Power, Gentofte, Denmark.

of critical, latency-sensitive data, essential for offshore wind farm operations. It is, therefore, crucial to design a dependable SDN-enabled IIoT-Edge network capable of running optimally amid the highlighted stochastic failures.

According to the IEC61907:2009 and IEC62673:2013 standards, a communication network is considered *dependable* if it remains reliable, available, maintainable, recoverable, and performs according to the stipulated industry service level agreements (SLA) [19], [20]. In this context, reliability is the probability that the network functions correctly under certain conditions over a specified time interval [21], [22], availability defines the readiness for network component usage at an instance of time, or the network up-time [23], maintainability is the probability of performing a successful repair action on the system within a given time under stated conditions [24], [25], recoverability measures the network's ability to bounce back from unexpected issues while preserving functionality [26], and performability refers to the ability to observe the variation in network service quality [27]. Given the criticality of the SDN-enabled IIoT-Edge network in facilitating latency-sensitive data transfer amid stochastic failure, this paper investigates the *dependability* of an offshore wind farm SDN-enabled IIoT-Edge network amid the highlighted stochastic failures and proposes novel detection, mitigation, and repair strategies.

Previous studies have evaluated system dependability in high availability and high-performing application scenarios using analytical and probabilistic assessment methods. Though no universal method exists for selecting the most appropriate dependability analysis technique, a verification process is employed to ascertain whether a given method aligns with the specified requirements. Nguyen et al. [28] use reliability graphs, a low accuracy, low model complexity analytical method, to determine the reachability of hosts amid stochastic failures. Xiao et al. [29] use fault tree analysis (FTA), a low accuracy, low model complexity analytical method to assess the reliability of a digital substation network by defining the minimal path set where failed components could result in overall system failure. FTAs provide a top-down approach to analyze the system's sub-components that cause system failure but fail to represent time or sequence dependency of events correctly [30]. To address the low accuracy concerns of FTAs and reliability graphs, researchers utilize Continuous-time Markov Models (CTMM) because of their demonstrated high accuracy levels in modeling the dynamic behavior of systems with exponentially distributed rates [31]. More specifically, Homogeneous Continuous-Time Markov Models (HCTMMs) are commonly used to model repairable systems with uncertain elements [22], [32]. Case in point, Vizarreta et al. [33] design a data-driven dependability assessment framework that models abstractions for imperfect distributed control plane and interaction with the service plane. Mendiratta et al. [34] design a Symbolic Hierarchical Automated Reliability and Performance Evaluator (SHARPE)-based probabilistic model to address the reliability concerns of their proposed SDN network but fails to address multiple switch failures in their model. Nencioni et al. [35] assess the SDN backbone using a two-level availability Markov model. Longo et al. [36] present a stochastic Markovian model that assesses the overall dependability of a hierarchically modeled SDN controller cluster. Moazzeni et al. [37] propose a reliable distributed SDN model that assesses the proposed network's reliability under several failure conditions. While CTMMs have demonstrated high accuracy levels, they fail to cater to large and complex systems because of the exponential growth in the number of states [38]. Combining CTMMs with stochastic Petri nets (SPNs) solves this challenge by defining models with fewer elements to represent a complex system.

B. Paper contributions

Based on the identified research gaps above, this paper delivers the following contributions:

- A novel IT/OT architecture for next-generation offshore wind farm data acquisition systems leveraging Industry 4.0 IIoT, SDN, edge computing, and virtualization technologies.
- A system model abstraction that maps stochastic failures in the control and data planes at a component level using stochastic Petri nets and defines an effective monitoring-agent-based strategy to detect, recover, and maintain an SDN-enabled IIoT-Edge network with a minimum number of redundant elements.
- A transient behavior analysis using a proof-of-concept simulation test bed to (i) estimate the impact of stochastic failures on the network service quality and (ii) estimate system parameters such as detection, repair, and service times.
- A steady-state behavior analysis using a probabilistic HCTMM to characterize the proposed system model's dependability in the long run recommending suitable network dependability assurance strategies and practices.

C. Organization of the paper

The remainder of the paper is organized as follows: Section II presents a novel IT/OT architecture for next-generation offshore wind farms and discusses the adoption of vendor-agnostic SDN to manage the resulting IIoT-Edge network. In Section III, the proposed network's system model is defined, highlighting stochastic failures in the control and data planes, along with an effective monitoring-agent-based strategy for detecting, recovering, and restoring the network to full operational state amid stochastic failures. Section IV outlines the methodology for evaluating the proposed network system model's transient and steady-state behavior using a proof-of-concept simulation test bed and a probabilistic HCTMM dependability assessment model, respectively. Section V presents and discusses the results, while section VI concludes the paper.

II. SDN-ENABLED IIOT-EDGE NETWORKS FOR NEXT-GENERATION OFFSHORE WIND FARMS

A. Application Scenario

Figure 1 illustrates a wide array of IIoT sensors deployed with K-out-of-N redundancy to acquire highly granular data samples from a fleet of wind turbines' critical infrastructure

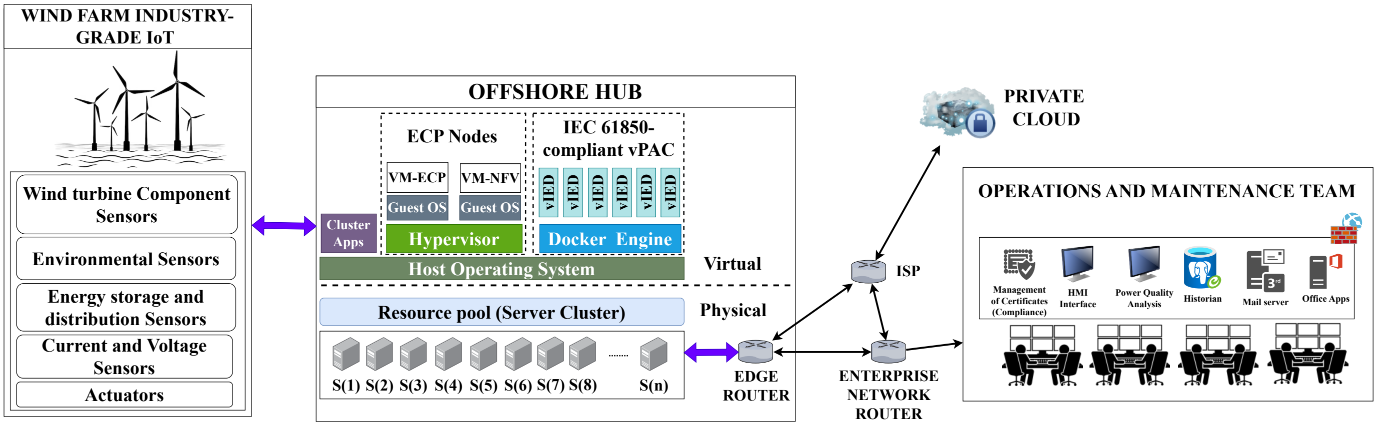


Fig. 1. A novel IT/OT architecture for next-generation offshore wind farms leveraging Industry 4.0 technologies such as IIoT, edge computing, virtualization, and vendor-agnostic software-defined networking to facilitate operation and maintenance functions.

[39]. The K-out-of-N redundancy ensures that the remaining sensors can continue capturing data samples if any sensor fails, maintaining continuous operation. These wind turbines are constructed around an offshore hub. Within the offshore hub, the digital substation rearchitectured as a data center hosts the edge computing platform (ECP) and the IEC61850-compliant virtual Protection, Automation, and Control (vPAC) nodes alongside other IT/OT infrastructure needed to support the wind farm operation and maintenance effectively. This data center provides sufficient computing, storage, and network resources through a unified resource pool. The resource pool provides a flexible and dynamic way to organize and aggregate multiple x86 servers' computing and memory resources. Its cluster applications provision the resources to the ECP and vPAC nodes based on policies set by the system administrators. The offshore hub's data center links to the remote O&M team's enterprise network via a wide area network (WAN) between the edge router, enterprise network router, and the Internet Service Provider (ISP) router, as shown in Figure 1. Using this IP/MPLS WAN backhaul, the O&M team can remotely commission calendar-based, and more recently condition-based tasks such as overall equipment inspection, replacement and repair of faulty components, and recalibration of IIoT sensors and actuators [40]. Zooming in on the link between the wind farm Industry-grade IoT and the offshore hub modules in Figure 1, an out-of-band control SDN-enabled IIoT-Edge network for the wind farm's field area is designed as illustrated in Figure 2. The ethernet switches connected to the server rack's physical network interface cards (NIC) conduct hardware-level filtering and checksum verification on the data samples at the rate of $\leq 10^{-9}s$ [41]. This data is handed over to the server cluster pool's host operating system, which runs the hypervisor and docker engine. The cluster application policies manage the virtual switch (vSWITCH) communication. Subsequently, the host operating system forwards the data streams to either the hypervisor's or the docker engine's vSWITCH network stack. The hypervisor's vSWITCH receives the data streams and forwards them to the MQTT broker while the docker engine forwards the data streams to the appropriate docker container's network configuration. The server cluster vSWITCH's internal communication is handled by its virtualization layer.

Two scenarios are defined to demonstrate the flow of sensor data from the wind turbines to the offshore hub data center's ECP and vPAC nodes and subsequent actuation as follows:

IIoT-to-ECP node scenario: The wind turbine IIoT sensor data samples are sent to the ECP nodes situated in the offshore hub through high-speed, low-latency communication sub-sea fiber optical cables to guarantee swift response [9], [10]. Periodically, these data samples received at the offshore hub's ECP node are stored in local database servers or sent to the enterprise network database servers and the private cloud through the IP/MPLS WAN (see Figure 1) for forecasting, pattern recognition, predictive analytics, and visualization. Ordinarily, analog measurements collected by IIoT sensors mounted in the wind turbines are sent to the turbine's local data acquisition module [11] through fiber optic patch cords or hardwired copper connections. The module pre-processes the sensor data, executes local control loops, aggregates the measurements, and then sends the data samples on the nacelle switch. These measurements are sent to the ECP node's message broker. When the ECP node's message broker receives the sensor data: (i) the message broker publishes these sensor data to the relevant topics (see Figure 2) using the Message Queuing Telemetry Transport (MQTT) protocol, and (ii) the IoT apps subscribe to different topics to coordinate inter-turbine communication and data exchange.

IIoT-to-vPAC node scenario: Additionally, the current and voltage sensors collect analog data from the digital substation processes and send it to a Merging Unit (MU). The MU samples and digitizes the current and voltage measurements, adding time stamps before broadcasting them over the network as Ethernet-based IEC61850-9-2 Sampled Values (SV) frames [42]. The vIED containers hosted in the vPAC docker environment subscribe to the published SV frames [43]. These vIEDs are set with specific threshold values for each parameter reflecting the system's safe operating limits. When the SV readings exceed set thresholds to indicate a fault or an abnormality, the vIED sends IEC61850 Generic Object-Oriented Substation Event (GOOSE) message signals to actuators to coordinate response with other vIEDs.

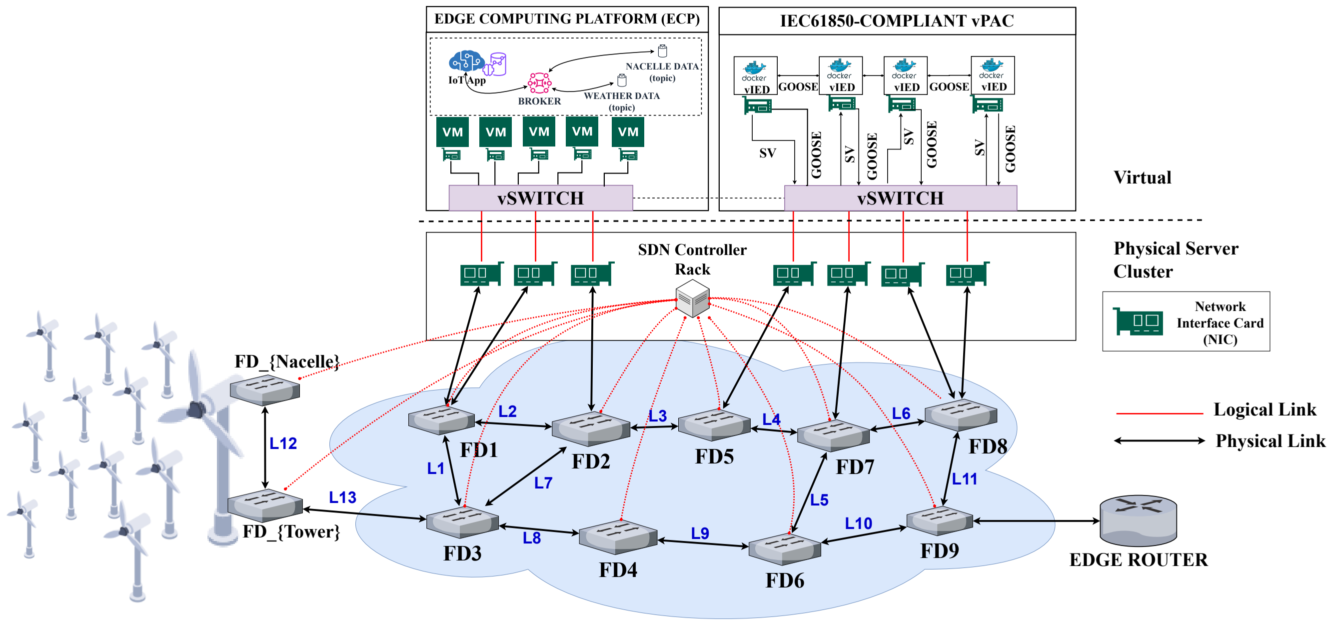


Fig. 2. An out-of-band control SDN-Enabled IIoT-Edge network schematic illustrating the fiber-optic-based connectivity between a fleet of wind turbine generator nacelle and tower switches connected to the offshore hub's data center switches $FD_{(1,\ldots,9)}$, leading to the server cluster's physical network interfaces and virtualized networks within the ECP and vPAC nodes.

B. Motivation for the adoption of software-defined IIoT-Edge network architecture in next-generation offshore wind farms

Conventional offshore wind farm communication networks adhere to the IEC61400-25 and IEC62439-3 standards that recommend specific communication protocols, information models, and a robust Ethernet-based network architecture [44] to facilitate the IIoT-to-ECP and IIoT-to-vPAC data transfer. Current deployments utilize the IEC 62439-3's High Availability Seamless Redundancy and Parallel Redundancy Protocols (IEC62439-3 HSR and PRP) to guarantee high availability and uninterrupted operation and redundancy [45]. By design, HSR sends duplicated Ethernet frames over two separate paths in a ring topology. Each node on the ring topology has two ports that the HSR uses to create and transmit these duplicate frames ensuring delivery despite potential failures. Conversely, PRP uses two parallel star topology networks (LAN A and LAN B), sending duplicate frames across both. The receiver processes the first frame and discards the duplicates, ensuring continuous data transfer even if one LAN fails.

1) IEC62439-3 HSR and PRP limitations: While there are a myriad of benefits to using the IEC62439-3 HSR and PRP, the technology faces several challenges. Firstly, scaling IEC62439-3 HSR and PRP implementations makes the network very complex to manage or maintain as it increases the task of implementing diverse configurations and keeping track of numerous functions [45], [46]. Secondly, scaling accrues substantial capital and operational costs indicating a growing financial burden on the wind farm operators [47]. For instance, as additional redundant switches, cabling, and networking components are required, infrastructure costs rise. Moreover, design, installation, and configuration (including consultancy and labor) expenses contribute to operational overheads. Thirdly, IEC62439-3 HSR and PRP are not suited for geographically dispersed networks such as the wind farm's IIoT-Edge network, because their design, which duplicates data traffic for redundancy, demands higher bandwidth. Lastly, the IEC62439-3 HSR and PRP networks lack inherent flexibility for dynamic changes and face interoperability issues, complicating data transfer across diverse network segments and technologies.

2) SDN offerings: To address the limitations of IEC62439-3 HSR and PRP, a fusion of OpenFlow-enabled IEEE802.1 Time Sensitive Networking (TSN) Ethernet switches referred to as Forwarding Devices (FDs) and vendor-agnostic Software-Defined Networking (SDN) are adopted in the IIoT-Edge network, building on the ideas in [18], [47] because of the following merits: Firstly, the SDN controller's global view of the network enables rapid updates to configurations and data paths for traffic management and fault failover, in $\leq 100\mu s$, compared to the conventional communication networks which take $10-30ms$ to converge [48]. Secondly, because the network is centrally controlled, it automates network management, proactively enforces intricate network policies, and adjusts resources in real-time for prioritized and critical data traffic [49]. Thirdly, while scaling the data plane in the SDN architecture results in increased flow initiation rates [50], this constraint is mitigated by distributing the workload within the controller cluster.

C. SDN-enabled IIoT-Edge network design specifications

Given the crucial role in enabling real-time, latency-sensitive data exchange, the proposed network is fortified through several design specifications. The network's hardware components are ruggedized to withstand the harsh and unpredictable environmental changes in the offshore context. Further, cost-effective redundant paths are built using link

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2024.3458447

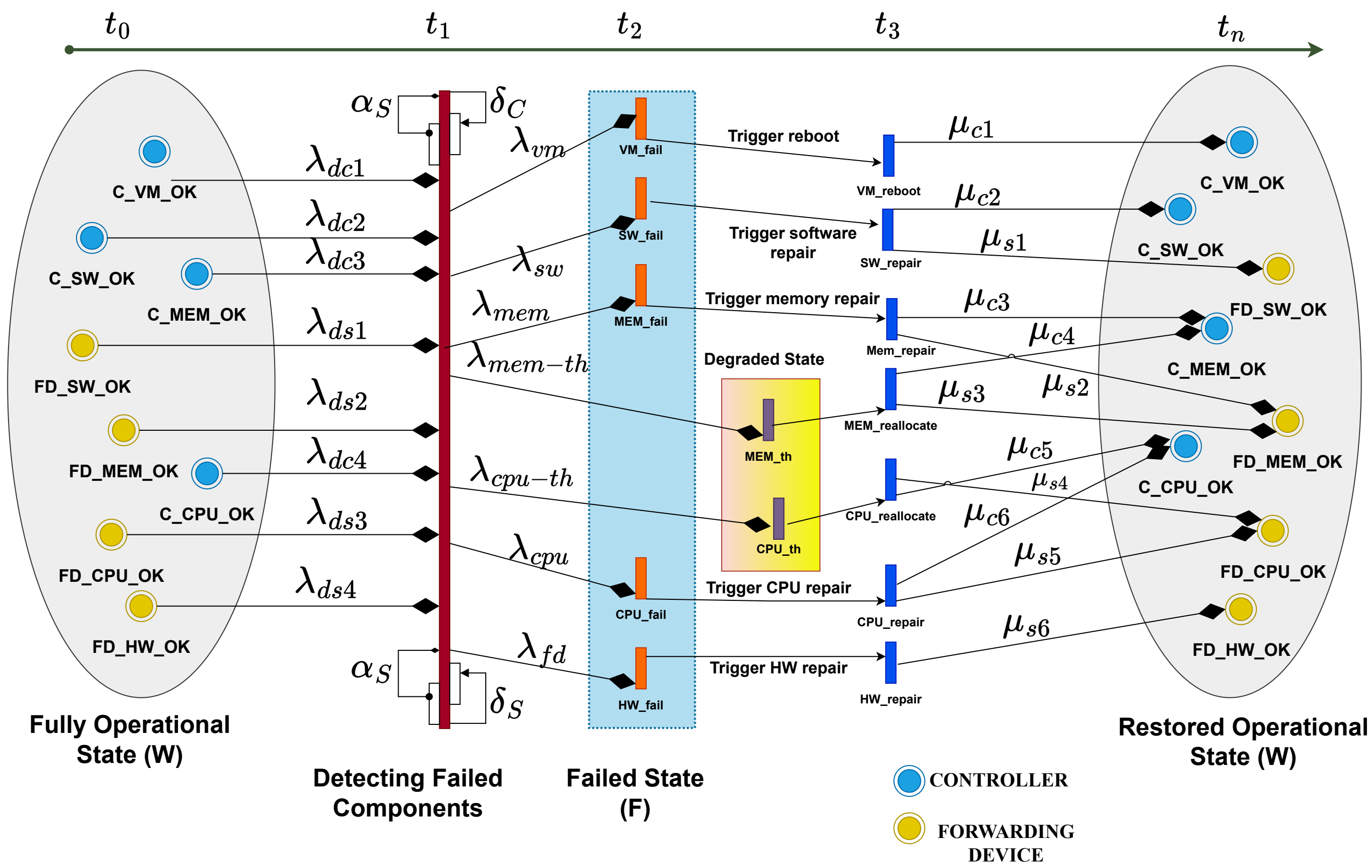


Fig. 3. A model abstraction that maps stochastic failures to a controller and forwarding device at individual component level to the network level and employs an effective strategy to detect, recover, and maintain the proposed network to a fully operational state using stochastic Petri nets.

aggregation techniques such as the EtherChannel Technology. The EtherChannel technology bundles several physical cables into a single logical link between the FDs. This technology improves redundancy and bandwidth and enhances fault tolerance in FD-to-FD connections [51], [52]. The SDN-controlled FDs facilitate dynamic reconfiguration to adapt to changing conditions and prioritize critical monitoring data and control over less critical network traffic. Periodic software upgrades and patches are deployed on all the virtual components to fix software-related bugs. Moreover, strong authentication and access control as well as stream ciphers are adopted to protect the critical data exchange from intruders.

The network operates on the cluster coordination fail-over approach where the SDN controllers in the control plane select a leader using the Raft consensus algorithm [53]. The leader contains the entire network's state information, makes decisions, and coordinates actions within the controller cluster [54]. The leader sends heartbeats or TCP keep-alive messages to the followers to share the current network state. This cluster coordination is necessary for the seamless migration of FDs when the primary controller fails. When a leader fails, the built-in fail-over logic detects the failure and triggers a process that migrates the FD connections to its followers. Notably, the SDN controller uses a high polling rate on a separate management network, in the out-of-band control mechanism, to collect and send information to the FDs through asynchronous OpenFlow messages.

III. System Model

The SDN-enabled IIoT-Edge network for a wind farm's field area network (FAN) is depicted in Figure 2. It features n_C controllers (configured as virtual machines inside the SDN controller server rack) each managing a virtual segment of the FAN and n_S FDs to represent the FD network of each segment. By design, each controller manages a single virtual segment and is clustered with n_C controllers serving as backups. Let $\mathcal{C}$ represent the set of SDN controllers in the control plane, $\mathcal{S}$ represent the set of FDs in the data plane, and $\mathcal{L}$ represent the set of links between the FDs. Without loss of generality, we model an SDN network architecture consisting of 3 controllers, 9 FDS, and 13 Links as illustrated in Figure 2.

A. Stochastic failure model abstraction mapping

Stochastic failures impact both controllers and forwarding devices (FDs) at the component level. This includes (i) random failures of virtual machine (VM) instances and software (SW) instances at the control plane, and (ii) random failures of the hardware (HW) components of FDs and their software, as well as CPU consumption above the threshold, and memory (MEM) leakages. As defined in [25], [29], [33], these control and data plane component failure rates, mean time to failure (MTTF) and mean time to repair (MTTR) are denoted in Table I. Given

TABLE I
CONTROL AND DATA PLANE COMPONENT FAILURE RATE [25], [29], [33]

Component	Failure Rate	MTTF(h)	MTTR(h)
λ_{cpu}	36.378×10^{-6}	27500	24
λ_{mem}	36.378×10^{-6}	27500	24
$\lambda_{cpu_{th}}$	7.205×10^{-7}	1400000	24
$\lambda_{mem_{th}}$	7.205×10^{-7}	1400000	24
λ_{vm}	16.67×10^{-6}	60000	24
λ_{fd}	5×10^{-6}	200000	24
λ_{fiber}	14.28×10^{-6}	70000	24
λ_{sf}	7.504×10^{-6}	134000	24

these stochastic failures, a model abstraction (see Figure 3) demonstrates an effective strategy to detect the highlighted stochastic failure, recover, and maintain the network to a fully operational state following failure.

1) At the control plane: Starting from the fully operational state, $\mathcal{W}$, the SDN controller's components (VM, SW, MEM, and CPU) can fail. A monitoring agent, running at the application plane, periodically checks the status of the network components and detects failure in any or all of the components registered in the switch's group and meter table metrics. This monitoring agent interacts with the control plane components through RESTful APIs reading metrics from the control plane's topology and statistics manager. The model transitions from working state to detection state at the rate λ_{dc_i} where $i = 1, ..., n$ number assigned to the controller components being monitored. The detection state, det_C, is a timed event such that if the model fails to transition to a failed state at a rate, λ_{vm_i} within a stipulated time (in this case 4 seconds - the heartbeat rate), it re-initiates failure detection at the rate, δ_{C_i}. Then, it determines the failed state using transition rate, α_{C_i}. The system transitions to the failed state ($\mathcal{F}$) and the monitoring agent triggers repair at the rate, μ_{C_i}. Regardless of the working state of the components in the data plane, when any of the highlighted controller components fails, the system automatically migrates (state mig_s) the FDs under that controller's administrative domain to another controller that is in its fully operational state at the rate, λ_m. However, it can also migrate these FDs to a controller in a degraded state if a working controller is unavailable.

2) At the data plane: When any FD components (VM, software, memory, and CPU) fail, the monitoring agent detects and transitions the FD to a failed state. The detection state is a timed event such that if the model fails to transition to a failed state at rate, λ_{fd_i}, within a stipulated time ($\cong 60s$), it re-initiates failure detection at the rate, δ_{S_i}. Then, it determines the failed state using transition rate, α_{S_i}. Its primary controller detects the failure when it no longer gets an acknowledgment to an *"OFPT-FLOW-MOD"* message from that particular FD. The controller reads the network state and uses the path computation element protocol (PCEP) running in the application plane to determine a new path to transfer the data streams. The monitoring agent triggers repair of the failed FD component at the rate, μ_{S_i}. When the FD is repaired and powered on, it reconnects to its primary controller to receive

TABLE II
SYSTEM LEVEL MODEL CONDITIONS

Conditions	C1	C2	C3	S1	S2	S3	S4	S5	S6	S7	S8	S9
$\mathcal{R}_1$	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
$\mathcal{R}_2$	✓	×	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
$\mathcal{R}_2$	✓	✓	×	✓	✓	✓	✓	✓	✓	✓	✓	✓
$\mathcal{R}_2$	×	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
$\mathcal{R}_3$	✓	✓	✓	✓	✓	×	✓	✓	×	✓	×	✓
$\mathcal{R}_4$	×	✓	×	✓	✓	✓	✓	✓	✓	✓	✓	✓
$\mathcal{R}_4$	✓	×	×	✓	✓	×	✓	✓	×	✓	×	✓
$\mathcal{R}_4$	×	✓	×	✓	✓	×	✓	✓	×	✓	×	✓
$\mathcal{R}_4$	×	×	✓	✓	✓	×	✓	✓	×	✓	×	✓

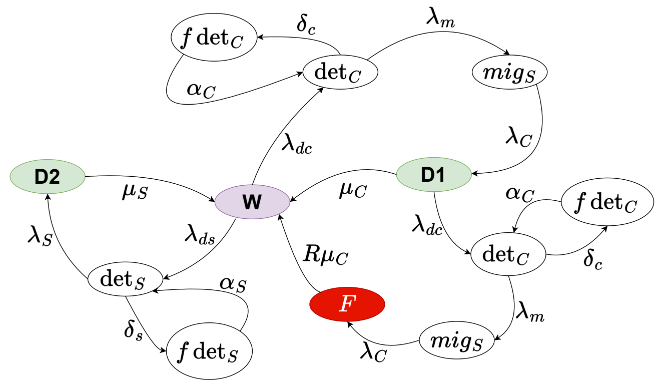


Fig. 4. The system-level Homogeneous Continuous Time Markov Model (HCTMM) is scaled from the model abstraction in Figure 3 using the model state conditions.

flow instructions and continue forwarding packets. From the model abstraction, both controller and FD components enter a degraded state when their memory and CPU consumption exceed the threshold. These components continue to function but not at an optimal level.

B. System-level homogeneous continuous time Markov model (HCTMM)

Scaling the model abstraction (see Figure 3) that only features a single controller and FD to a larger network (see Figure 2), more elements are considered. To simplify the resulting model abstraction stochastic Petri net, an HCTMM is defined to capture all the resulting transitions using a three-state ($\mathcal{W}, \mathcal{D}, \mathcal{F}$) system model. As such, $\mathcal{W}$ is the working state, $\mathcal{D}$ are the degraded states, and $\mathcal{F}$ is the failed state as illustrated in Figure 4. At the system level, when all controllers' and FDs' components are fully operational, the network model is in $\mathcal{W}$. When these components at both the control and data plane are degraded, then the network model transitions to a degraded state, $\mathcal{D}$. Lastly, when all or some components are in the failed state, the network model transitions to the failed state, $\mathcal{F}$.

1) Model conditions: Considering the interaction between the control and data plane, some conditions are applied to the scaled model abstraction as denoted in Table II,

- $\mathcal{R}_1$: All controllers, $\mathcal{C}_{(1,..,3)}$ and all FDs $\mathcal{S}_{(1,..,9)}$ are fully operational.
- $\mathcal{R}_2$: 2 out of 3 controllers, $\mathcal{C}_{(1,2)}$ OR $\mathcal{C}_{(2,3)}$ OR $\mathcal{C}_{(1,3)}$, are working, all FDs, $\mathcal{S}_{(1,...,9)}$, are fully operational.

TABLE III
TRANSITION RATE MATRIX

$$
\mathcal{Q} = \begin{pmatrix}
-(\lambda_{dc}+\lambda_{ds}) & \lambda_{dc} & 0 & 0 & 0 & \lambda_{ds} & 0 & 0 & 0 \\
0 & -(\delta_C+\lambda_M) & \delta_C & \lambda_M & 0 & 0 & 0 & 0 & 0 \\
0 & \alpha_C & -(\alpha_C) & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -2(\lambda_C) & \lambda_C & 0 & 0 & 0 & \lambda_C \\
\mu_C & \lambda_{dc} & 0 & 0 & -(\mu_C+\lambda_{dc}) & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -(\delta_S+\lambda_S) & \delta_S & \lambda_S & 0 \\
0 & 0 & 0 & 0 & 0 & \alpha_S & -(\alpha_S) & 0 & 0 \\
\mu_S & 0 & 0 & 0 & 0 & 0 & 0 & -(\mu_S) & 0 \\
R(\mu_C) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -R(\mu_C)
\end{pmatrix}
$$

- $\mathcal{R}_3$: All controllers, $\mathcal{C}_{(1,..,3)}$, and some of the FDs $\mathcal{S}_{(1,2,4,5,7,9)}$ are fully operational, assuming FDs, $\mathcal{S}_{(3,6,8)}$, are down.
- $\mathcal{R}_4$: When only one controller: $\mathcal{F} = (\mathcal{C}_1$ or $\mathcal{C}_2$ or $\mathcal{C}_3)$ is working regardless of the number of operational FDs.

2) Model assumptions: Several assumptions are made in this system model,

- Links $\mathcal{L}_{(1,...,13)}$ are configured using the EtherChannel link aggregation technology with a mean time before failure (MTBF) of 70,000 hours ($\lambda_{fiber} = 14.28 \times 10^-6)$ as in Table I. Therefore, this model assumes negligible link failure.
- Further, this model assumes negligible FD-to-controller latency as the SDN controller cluster uses a distributed approach to handle increased flow initiation requests.

Conventionally, the system model's HCTMM is denoted as a tuple $\langle \mathcal{M}, \mathcal{M}_{init}, \mathcal{Q} \rangle$ where $\mathcal{M}$ is a finite set of states, $\mathcal{M}_{init}$ is the initial state, $\mathcal{Q} \to (\mathcal{M} \times \mathcal{M})$ is the transition rate matrix [31]. The resulting HCTMM, as shown in Figure 4, comprises nine distinct states namely

$$\mathcal{M} = \{\mathcal{W}, \mathcal{D}1, \mathcal{D}2, \mathcal{F}, detC, fdetC, detS, fdetS, migS\}$$

where the time spent in each state is exponentially distributed [55].

$$\mathcal{M}_{init} = \{\mathcal{W}\} \to \text{fully operational state}$$

The proposed HCTMM is irreducible, allowing access to all states via intermediate states, as shown in Figure 4, which depicts it as a single communicating class. Furthermore, the HCTMM is ergodic, with positive recurrence, ensuring finite expected return times to any given state, and as such it converges to a unique stationary distribution regardless of the initial state. This ensures computational tractability when modeling large dynamic systems. The transition rate matrix, $\mathcal{Q}$, assigns time-independent transition rates to each pair of states as $q[i][j]_{\forall(i \neq j)}$ to represent the rate of going from state i to state j as illustrated in Table III. In this transition matrix, $\mathcal{Q}_{ij} = 0$ for states without direct transition. Additionally, $\mathcal{Q}_{ii}$ is the transition rate matrix's diagonal entry denoted by negative values that equate to the sum of the rates out of state i to all other states such that $\mathcal{Q}_{ii} = -\sum_{i \neq j} q_{ij}$. The timed transition rates (δ, α, λ_m, λ_s) for the HCTMM of the proposed system are determined using eqn. 1,

$$\delta \text{ or } \lambda_m = -\frac{ln(1-P)}{T} \tag{1}$$

TABLE IV
HCTMM MODEL PARAMETERS

Parameter	Baseline [28], [34], [35]	Test Scenarios
Failure Rates		
$(\lambda_C) \to$ Controller failure rate	50/year	[1-100]/year
$(\lambda_S) \to$ Switch failure rate	25/year	[1-100]/year
Detection Rates		
$(\lambda_{dc}) \to$ Controller failure detection rate	3600/hour	[1000-5000]/hour
$(\lambda_m) \to$ Switch migration rate	2074/hour	[1000-10000]/hour
$(\lambda_{ds}) \to$ Switch failure detection rate	3600/hour	[10000-50000]/hour
Repair Rates (from Section IV-A)		
$(\mu_S) \to$ Switch add/repair rate (offline repair)	25/hour	[1-100]/hour
$(\mu_C) \to$ Controller add/repair rate (offline repair)	32/hour	[1-100]/hour
Intermediate States (Timed Events)		
$(\delta_C\ ) \to$ detC to fdetC	9/hour [P=0.01]	P=[0 - 0.99]
$(\alpha_C\ ) \to$ fdetC to detC	4145/hour [P=0.99]	P=[0 - 0.99]
$(\delta_S\ ) \to$ detS to fdetS	9/hour [P=0.01]	P=[0 - 0.99]
$(\alpha_S\ ) \to$ fdetS to detS	4145/hour [P=0.99]	P=[0 - 0.99]
$\mathcal{R} \to$ **Coverage factor = 0.97**		

where $(1 - P)$ is the probability that the system does not transition within the stipulated time, T. For instance, $detC$ should transition to $migS$ within 4 seconds with a probability of 0.99, else to $fdetC$ with a probability of 0.01 for a system with efficient fail-over. This implies that, $\lambda_m = 2074$ transitions/hour and $\delta_C = 9$ transitions/hour as denoted in Table IV.

The repair occurs from states, $\mathcal{D}1, \mathcal{D}2$, and $\mathcal{F}$ back to the fully operational state, $\mathcal{W}$ as in eqn.2,

$$\mu_{C_{hr}} \text{ or } \mu_{C_{ar}} = \begin{cases} R\mu_C & \text{for } \mathcal{F} \to \mathcal{W} \\ \mu_C & \text{for } \mathcal{D}_1 \to \mathcal{W} \\ \mu_S & \text{for } \mathcal{D}_2 \to \mathcal{W} \end{cases} \tag{2}$$

where $\mu_{C_{hr}}$ is the repair with a human-in-the-loop and $\mu_{C_{ar}}$ is the repair with an automated script.

Further, the Chapman-Kolmogorov Forward Equations denoted in eqn. 3 and eqn. 4 are used to compute the steady-state probabilities which indicate the long-term likelihood of the system remaining in each state,

$$\pi'(t) = \pi(t)\mathcal{Q} \tag{3}$$

$$\pi\mathcal{Q} = 0 \text{ subject to} \sum \pi_i(t) = 1 \tag{4}$$

where $\pi(t)$ denotes the steady-state probability distribution and $\mathcal{Q}$ denotes the transition rate matrix.

For the n-state HCTMM, the steady-state probability vector given as $\pi = [\pi_1, \pi_2, ..., \pi_n]$ satisfies $\pi\mathcal{Q} = 0$. It is expressed

TABLE V
PHYSICAL SERVER AND VIRTUAL MACHINE SPECIFICATIONS

Specifications	VM-A	VM-B	VM-C	VM-D
SDN Architecture	Control Plane			Data Plane
Hard Disk Space	60GB	60GB	60GB	128GB
Processor	Intel®Xeon®			
Hypervisor	Linux Kernel-based Virtual Machine (KVM) Docker engine version 23.0.9			
CPU, RAM	12GB, 20 Cores@3.60GHz			

as a system of linear equations in eqn 5,

$$\begin{cases} \pi_1 q_{11} + \pi_2 q_{21} + \cdots + \pi_9 q_{n1} = 0 \\ \pi_1 q_{12} + \pi_2 q_{22} + \cdots + \pi_9 q_{n2} = 0 \\ \vdots \\ \pi_1 q_{1n} + \pi_2 q_{2n} + \cdots + \pi_n q_{nn} = 0 \end{cases} \tag{5}$$

A normalization condition whose steady-state probabilities sum to 1, as in eqn.6, is added to the current system of linear equations

$$\pi_1 + \pi_2 + ... + \pi_n = 1 \tag{6}$$

Based on the steady-state probabilities, the operational states of the proposed model are $\pi_{(oper)} \rightarrow (\mathcal{W}, \mathcal{D}_i)$ and $\pi_{(failed)} \rightarrow (\mathcal{F})$. From the steady-state probabilities of the operational states $\pi_{(oper)}$, the availability of the proposed system HCTMM is given as,

$$\mathcal{A} = \sum_{i=1}^{m} \pi_{(oper)_i} \tag{7}$$

where i to m are the operational states.

Similarly, all the transition rates from the operational states to the failed state are summed to assess the expected time until the system fails for the first time.

$$MTTF_{(sys)} = \frac{1}{\sum_{i=m+1}^{n} \sum_{j=1}^{m} q_{ij} \pi_i} \tag{8}$$

IV. A Two-Pronged Approach to Dependability

This section presents a two-pronged approach[1] used to investigate the dependability of the SDN-enabled IIoT-Edge network in the offshore wind farm application scenario. To begin with, a proof-of-concept simulation testbed observes the transient behavior of this network amid the highlighted stochastic failures. Further, a quantitative assessment is conducted to determine the network's steady-state behavior amid stochastic failure using the HCTMM approach introduced in the previous section.

A. Transient behavior analysis

On a physical server, three VMs were instantiated for the 3 SDN controllers, such that VM-A hosted C_1, VM-B hosted C_2, and VM-C hosted C_3. Table V denotes the server and virtual machine specifications for the proof-of-concept simulation test bed as illustrated in Figure 5. The OpenDaylight SDN controller (ODL SDNC) is used in this setup. ODL SDNC has a distributed-flat architecture which means that the SDN controller cluster comprises SDN controllers interacting as peers, unlike the hierarchical architecture where some SDN controllers are head controllers overseeing sub-level controllers [56]. In the ODL SDNC, *"akka.conf"* and *"modules-shards.conf"* configuration files are modified to define the cluster node members and associated member links and ports [33], [57], [58]. The network's data plane comprising wind turbine and PDC FDs (see Figure 2) was implemented in a Mininet network emulator hosted in VM-D as illustrated in Figure 5. The data plane FDs remotely connect to an SDN controller cluster running on VM-A, VM-B, and VM-C using the southbound interface OpenFlow Ver-1.3 protocol.

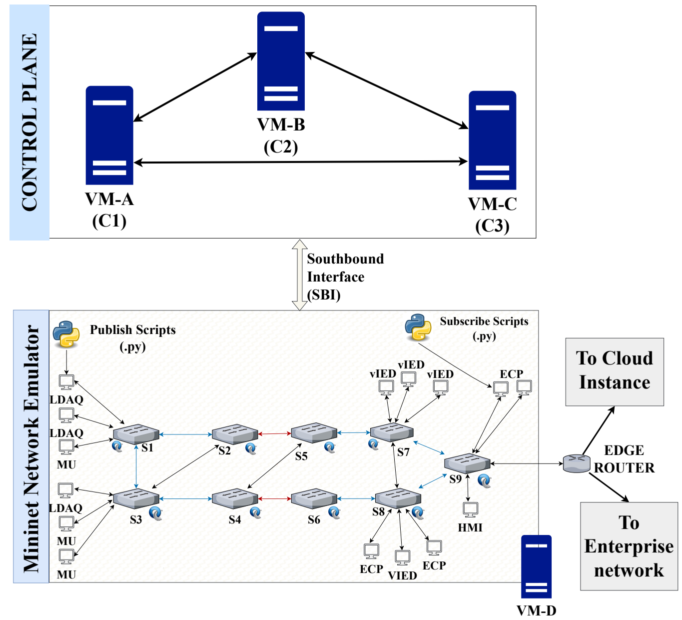


Fig. 5. Transient behavior analysis simulation testbed where LDAQ is the local data acquisition module in the wind turbines, MU is the merging unit, vIED is a virtual intelligent electronic device in the vPAC node and the ECP represents the IoT apps in the ECP node.

To simulate the IoT-to-ECP nodes scenario (see section II-A), a publish script was run on one of the virtual hosts connected to an FD in the network topology, and a separate subscribe script was run on another host within the same network. The publish script publishes IIoT sensor data (mimicking the wind turbine's data acquisition module). Similarly, the subscribe script (mimicking the IoT Apps) subscribes to the data sent to topics managed by the MQTT broker running on the same VM. To simulate the IoT-to-vPAC nodes scenario (see sectionII-A), a publish script based on LibIEC61850[2] open-source library was run on several virtual hosts to publish SVs data streams of current and voltage measurements to the Mininet's FD network. On a separate set of virtual hosts, a subscribe script (that mimics vPAC nodes' vIEDs) is run to subscribe to the already published SV streams. In different test cases, 2-3 concurrent streams of SV are published on the network as an example of normal operation for a single bus.

The impact on the network service quality, amid stochastic

[1]Source code will be provided upon acceptance.

[2]Open-source library for the IEC 61850 protocols.

TABLE VI
NORMALIZED FAILURE RATES, REPAIR RATES FOR 15 TEST CASE SCENARIOS

Item	Description	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7	Case 8	Case 9	Case 10	Case 11	Case 12	Case 13	Case 14	Case 15
λ_{dc}	[W to detC]	0.0000479	0.0000108	0.0000607	0.0000387	0.0000194	0.0000439	0.0000014	0.0000682	0.0000831	0.0000848	0.0000188	0.0000085	0.0000657	0.0000772	0.0000030
δ_C	[detC to fdetC]	0.0000098	0.0000093	0.0000001	0.0000076	0.0000066	0.0000059	0.0000083	0.0000082	0.0000072	0.0000041	0.0000025	0.0000072	0.0000039	0.0000038	0.0000065
α_C	[fdetC to detC]	0.0000010	0.0000001	0.0000004	0.0000004	0.0000004	0.0000010	0.0000004	0.0000003	0.0000002	0.0000005	0.0000005	0.0000010	0.0000009	0.0000004	0.0000003
λ_M	[detC to migS]	0.0000015	0.0000022	0.0000014	0.0000016	0.0000019	0.0000006	0.0000025	0.0000021	0.0000024	0.0000026	0.0000005	0.0000003	0.0000001	0.0000019	0.0000006
λ_{SS}	[migS to D1]	0.0000903	0.0000128	0.0000470	0.0000855	0.0000984	0.0000076	0.0000537	0.0000801	0.0000569	0.0000209	0.0000601	0.0000665	0.0000925	0.0000884	0.0000432
μ_C	[migS to F]	0.0045055	0.0013092	0.0213217	0.0092248	0.0122846	0.0089313	0.0022697	0.0136312	0.0085690	0.0276446	0.0174451	0.0099801	0.0079909	0.0043159	0.0179661
λ_{ds}	[D1 to W]	0.0000251	0.0000668	0.0000378	0.0000197	0.0000503	0.0000666	0.0000963	0.0000437	0.0000283	0.0000018	0.0000766	0.0000456	0.0000333	0.0000758	0.0000473
δ_S	[W to detS]	0.0000047	0.0000088	0.0000004	0.0000096	0.0000038	0.0000032	0.0000072	0.0000087	0.0000074	0.0000100	0.0000014	0.0000046	0.0000017	0.0000081	0.0000071
α_S	[detS to fdetS]	0.0000002	0.0000002	0.0000007	0.0000009	0.0000003	0.0000001	0.0000000	0.0000006	0.0000007	0.0000005	0.0000007	0.0000002	0.0000003	0.0000009	0.0000004
λ_S	[fdetS to detS]	0.0000011	0.0000015	0.0000029	0.0000003	0.0000008	0.0000028	0.0000026	0.0000021	0.0000004	0.0000018	0.0000008	0.0000009	0.0000012	0.0000012	0.0000021
μ_S	[detS to D2]	0.0231425	0.0175325	0.0167238	0.0214120	0.0193097	0.0085218	0.0276982	0.0037856	0.0115562	0.0186190	0.0159696	0.0182488	0.0178217	0.0093540	0.0055525
μ_C	[D2 to W]	0.0000006	0.0000018	0.0000017	0.0000005	0.0000011	0.0000013	0.0000017	0.0000022	0.0000001	0.0000016	0.0000015	0.0000022	0.0000002	0.0000019	0.0000010
$\mathcal{R}$	[F to W]	0.9500000	0.9500000	0.9500000	0.9500000	0.9500000	0.9500000	0.9500000	0.9500000	0.9500000	0.9500000	0.9500000	0.9500000	0.9500000	0.9500000	0.9500000

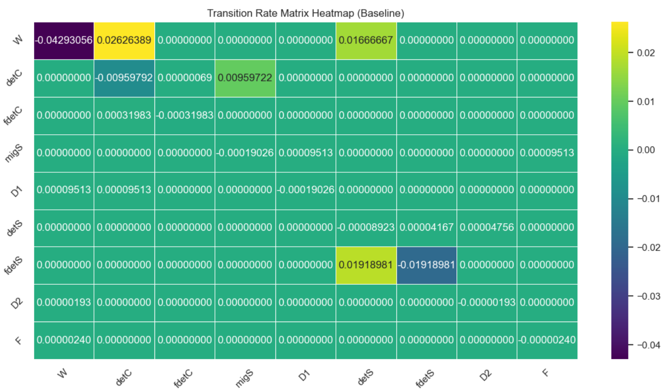


Fig. 6. The transition rate matrix heatmap populated using Table III and Table IV baseline values.

failures, was observed using the following metrics: throughput, packet loss, controller response time, and FD fail-over time on Wireshark®. Further, during the fault injection scenarios, the time taken for fail-over is measured to describe the performance deviation from normal behavior. Random controller and FD faults, software malfunctions, CPU overconsumption, and memory leakages were tested in the simulated network. Specifically, three tests were done: (i) changing the FD state in the data plane, (ii) introducing a controller component failure in the control plane cluster, and (iii) introducing packet loss at the rate of 1% and 10% on the most crucial paths to simulate degraded FD performance.

B. Steady-state behavior analysis

Having observed the transient behavior of the proposed network amid stochastic failure using the network throughput metric, the HCTMM (see section III-B) was implemented in PRISM® ModelChecker based on the component-level model abstraction (see Figure 3) and the system-level network conditions (See Table II). The states of the controller- and FD components were initialized, and the transition logic was adjusted to fit the specifications of a 3-controller cluster and a 9-FD data plane network topology. Using the system model baseline failure and repair rates from Table IV, the HCTMM was populated to generate a transition rate matrix, $\mathcal{Q}$, illustrated as a heatmap in Figure 6 featuring the baseline parameters.

Initially, baseline failure and repair rates were tested on the PRISM-based HCTMM, and simulations were run to determine the steady-state availability and system reliability. Further, 15 case scenarios based on different failure and repair rate combinations were defined within the range denoted in Table IV for the sensitivity analysis. To ascertain specific combinations for each case scenario, random scenarios were delineated according to the model abstraction mapping. For instance, Case 1 entailed an augmented failure of the controller VM alongside persistent memory leakages, with no failures occurring at the data plane. Conversely, Case 7 involved heightened failures in both the data plane hardware and software components, accompanied by sustained CPU overconsumption. Additionally, the effect of reducing the repair rate from the stipulated 24-hour MTTR (see Table I), holding the detection and migration rates constant, on steady-state availability and system reliability was examined. These decisions were undertaken to delineate diverse case scenarios; nevertheless, it should be noted that they may not comprehensively encompass all potential failure combinations. The steady-state behavior was observed by exporting the baseline and 15 case scenario data and solving steady-state probabilities using Python's *"scipy.linalg.null_space"* function for the modified $\mathcal{Q}$ matrix.

V. RESULTS AND DISCUSSION

The transient behavior of an unrepairable network model is illustrated in Figure 7 as an exponential distribution. To the left, the plot shows that the system takes 1450 hours to encounter the first complete failure within the 5000 hours of mission time, equating to a 29% network up-time. Considering that the system starts from a fully operational state, $\mathcal{W}$, this shows that without effective repair, the system can only be running 29% of the stipulated mission time which is significantly below the SLA threshold set at 99.999%[3] [59] network up-time underscoring a lack of dependability. As such, the proposed effective strategy for detecting, recovering, and maintaining the system is needed for a repairable network model. To observe the transient behavior of the proposed network amid stressed conditions, the system model's conditions in Table II were implemented in a proof-of-concept simulation test bed.

1) At the control plane: From the working state, $\mathcal{W}$, several stress tests were performed to monitor how the network responds to the failed controller(s). Using Wireshark®, as

[3] 99.999% network availability as stipulated in the industry service level agreements (SLAs) is referred to as 5 nines which implies that the network is down only 5 minutes and 15 seconds per year.

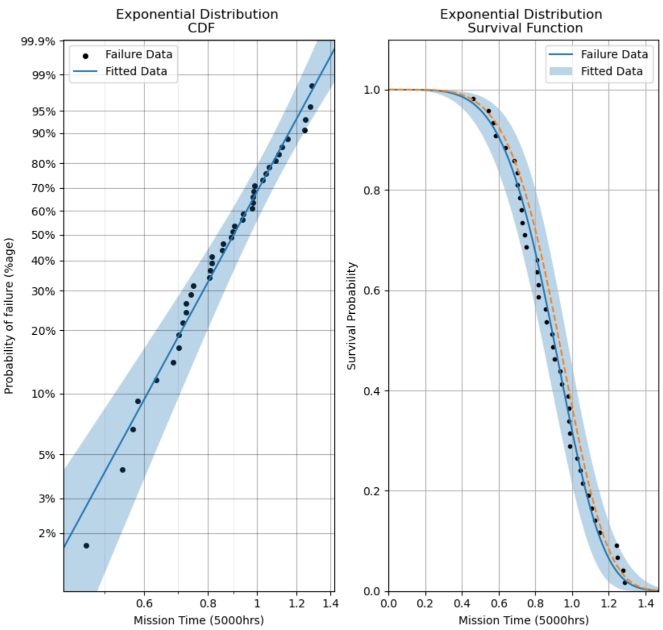


Fig. 7. The transient behavior of an unrepairable system model illustrating fitted failed data using an exponential distribution model within a 5000-hour of mission time.

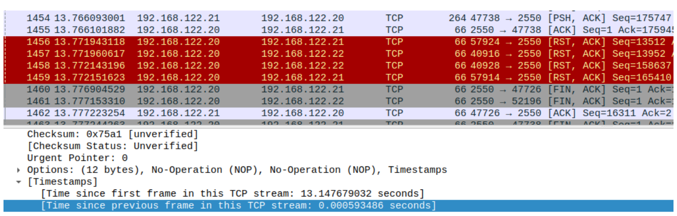
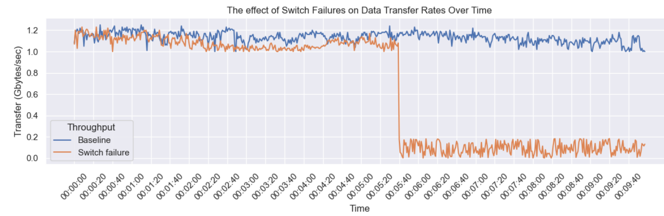
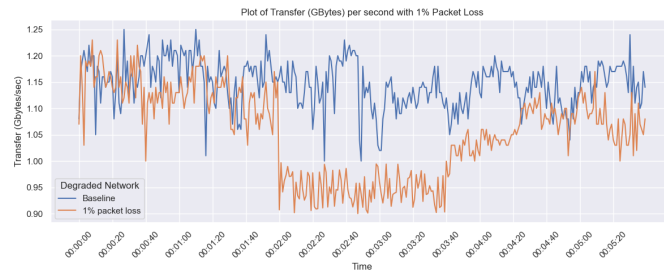


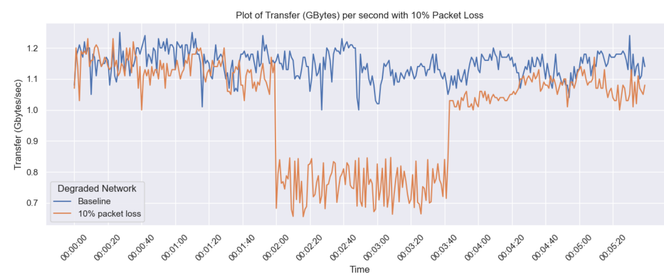
Fig. 8. The controller failure detected at $C_\lambda = 0.593$ ms, using Wireshark® for an SDN controller cluster with a 4 seconds heartbeat rate (HBR).



(a) Iperf3 network throughput before and after forwarding device(s) on a critical path fails.



(b) Iperf3 network throughput for a slightly degraded data plane network.



(c) Iperf3 network throughput for an adversely degraded data plane network.

Fig. 9. The proposed system model transient behavior measured using Iperf3 network throughput monitoring tool.

illustrated in Figure 8, it was observed that the other controllers took approximately 0.6 milliseconds (C_λ = 0.000593486 seconds) to detect that one of the controllers in the cluster had failed. Fundamentally, these controllers exchange heartbeat messages (every $4s$) through the East-West bound interface to confirm operational status and connectivity. The time to detect a controller failure is measured by the interval between TCP KeepAlive messages in Wireshark®. To determine the Mean Time to Detect (MTTD) for the simulated controller cluster (comprising $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3$), several failure tests were carried out

TABLE VII
TIME TO DETECT CONTROLLER FAILURE AS CAPTURED ON WIRESHARK®

C1	**Down**	**Up**	**Up**
C2	**Up**	**Down**	**Up**
C3	**Up**	**Up**	**Down**
Test I	0.000634460	0.000593486	0.000518461
Test II	0.000482294	0.000686359	0.000415660
Test III	0.000460635	0.000515686	0.000533394
Test IV	0.000294723	0.000593486	0.000415482
Test V	0.000418334	0.000634305	0.000395518
AveragePerTest	**0.000458089**	**0.00060404664**	**0.000455703**
MTTD=0.000505946213 $\cong$ 505.946 μS			

as denoted in Table VII. These tests involved deactivating one controller and observing the time taken by other controllers to detect the failure. The controller's Karaf log feature also documented the failure on a log sheet. Eqn. 9 computes the MTTD as an average of the time taken to detect controller failure (C_λ) over n tests,

$$C_{(MTTD)} = \frac{\sum_{i=1}^{n} C_{\lambda_i}}{n} \tag{9}$$

The MTTD for this 3-controller cluster was estimated to be $\cong 505.946\mu S$ as denoted in Table VII. From the simulation, it was observed that each controller in the controller cluster shared its network information with its peers, hence when one controller was lost, the other controllers took over administration of the FDs in the data plane. This was confirmed by checking if new flows were present in the FDs' flow tables after the initial flow expiry hard timeout ($FD_{MTTD} \cong 60$ seconds).

2) At the data plane: When an FD on a critical path in the network topology designed in the evaluation setup in Figure 5 fails, the SDN controller cluster attempts to find alternative paths to redirect traffic. IIoT-to-ECP and IIoT-to-vPAC publish/subscribe scripts were executed on multiple virtual hosts (*LDAQ, MU, ECP, vIED*) to simulate data exchange

TABLE VIII
STEADY-STATE PROBABILITIES FOR THE 15 CASE SCENARIOS

	W	**detC**	**fdetC**	**migS**	**D1**	**DetS**	**fdetS**	**D2**	**F**
Scenario-1	0.835348	0.015635	0.033076	0.010013	2.434093e-07	0.059393	0.04652	1.3e-05	3.00502e-06
Scenario-2	0.683357	0.059016	0.081102	0.144455	7.523393e-07	0.024633	0.007433	2e-06	1.905334e-06
Scenario-3	0.608196	0.039265	0.014967	0.025067	3.751728e-08	0.203373	0.109123	6e-06	3.329833e-06
Scenario-4	0.752306	0.056211	0.040331	0.036133	3.005067e-07	0.077328	0.03768	1e-05	9.699632e-07
Scenario-5	0.601651	0.037471	0.179774	0.026991	4.995114e-08	0.058366	0.095737	7e-06	3.239335e-06
Scenario-6	0.509643	0.075485	0.212136	0.108981	3.861437e-07	0.066929	0.026811	8e-06	6.445912e-06
Scenario-7	0.242107	0.431954	0.254912	0.025025	5.92799e-07	0.033042	0.012904	1.3e-05	4.279833e-05
Scenario-8	0.569143	0.100241	0.109708	0.047115	2.534061e-06	0.100314	0.07343	2.9e-05	1.840958e-05
Scenario-9	0.251691	0.026448	0.010867	0.026927	2.830005e-07	0.579376	0.104687	2e-06	2.651417e-06
Scenario-10	0.55141	0.040368	0.071035	0.013256	2.473096e-08	0.12419	0.199735	6e-06	9.022927e-07
Scenario-11	0.589836	0.020883	0.022898	0.0284	3.666146e-07	0.14612	0.191814	3.9e-05	1.022303e-05
Scenario-12	0.412051	0.065687	0.032698	0.012346	1.926607e-07	0.228595	0.248616	5e-06	1.698633e-06
Scenario-13	0.724266	0.06216	0.036936	0.108818	3.321703e-05	0.032043	0.0355	9e-05	0.0001545603
Scenario-14	0.710102	0.052509	0.067042	0.033722	5.936272e-07	0.039357	0.097255	6e-06	6.46769e-06
Scenario-15	0.357879	0.134116	0.225683	0.024392	5.623995e-07	0.209467	0.048451	7e-06	5.754386e-06

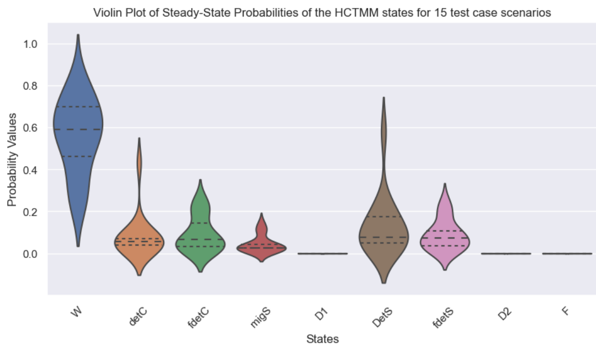
between wind turbines and the offshore hub as illustrated in the evaluation setup in Figure 5. Then, on two test virtual hosts on the extreme ends of the network topology, *Iperf3* commands were run to observe the network's throughput (per second) based on the data transmitted through the FD network. FD failure manifests in two different ways: (i) permanent damage where the FD is completely off or (ii) degraded FD state where the FD experiences a gradual decline in performance due to hardware aging or software issues [60]. Before stochastic faults were introduced in the data plane, the *iperf3* tool demonstrated stable network throughput as shown in Figure 9(a)'s baseline time series (in blue). Changing the FD's state from *"STATE UP"* to *"STATE DOWN"* to mimic permanent damage led to an initial throughput perturbation, followed by a cessation of data transfer, as shown in Figure 9(a)'s FD failure time series (in orange). Lastly, NetEm's [61] traffic control command was used to simulate both unexpected network traffic spikes and FD hardware/software degradation. Figure 9(b) and Figure 9(c) illustrate how this NetEm traffic control packet loss command introduces a 1% packet loss and a 10% packet loss on the path FDs degrading the network service quality by lowering the transfer throughput.

Implementing the effective monitoring-agent-based detection and recovery strategy, it was observed that the network restored to a fully operation state at 113 seconds ($\mu_{C_{hr}}$) in the manual restore and for the automated script, in 29 seconds ($\mu_{C_{ar}}$). Notably, this repair strategy adopted in the proof-of-concept test bed assumed negligible discrepancies in the automated reboot, script errors, or incorrect log analysis by the monitoring agent. Additionally, FD repairs were command-based and, hence, did not demonstrate an accurate assessment of physical repair times. This transient behavior analysis showed using the network throughput metric the extent of performance degradation during the transient periods.

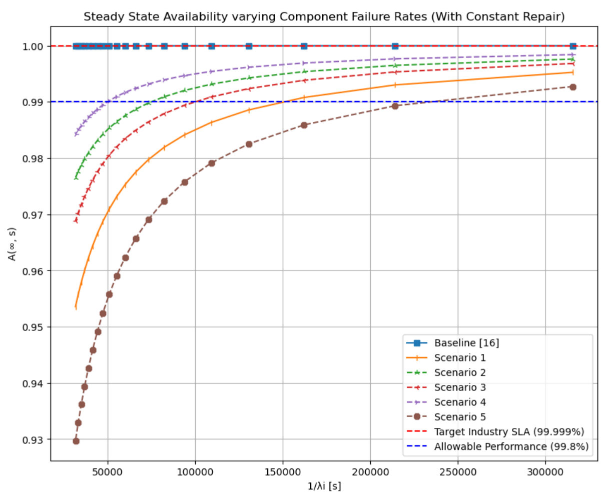
To assess long-term performance post-transient periods, an HCTMM, modeled on PRISM®, underwent simulation with a baseline and 15 test case scenarios. The objective of the proposed detection and recovery strategy is to achieve sustained operational status. Steady-state probabilities for both the baseline and the 15 test case scenarios were obtained as denoted in Table VIII. A violin plot illustrated in Figure 10(a) shows the steady-state probability distributions ranging from [0,1] on the y-axis against the system model states on the x-axis. Each violin contains a miniature box plot that uses a central mark to represent the median of the data and box extents to demonstrate the interquartile range with the 25th percentile below the median and the 75th percentile above the median. Notably, this violin plot has outlier data points attributed to the heavy-tailed distribution and variability in the derived steady-state probabilities.

From the violin plots in Figure 10(a), we see that the system likely remains working $\mathcal{W}$ amid stochastic failures, as indicated by the bulging region of the left-most violin plot in the probability range [0.4, 0.8]. The proposed model's detection, recovery, and maintainability strategy is notably effective, with minimal time spent in degraded and failed states $\mathcal{D}1$, $\mathcal{D}2$, and $\mathcal{F}$ because these states revert to $\mathcal{W}$ post-repair (see HCTMM transitions in Figure 4). The system transitions through intermediate states ($detC, fdetC, detS, fdetS, migS$) during controller or FD loss with a probability of up to 0.3 during stochastic failure.

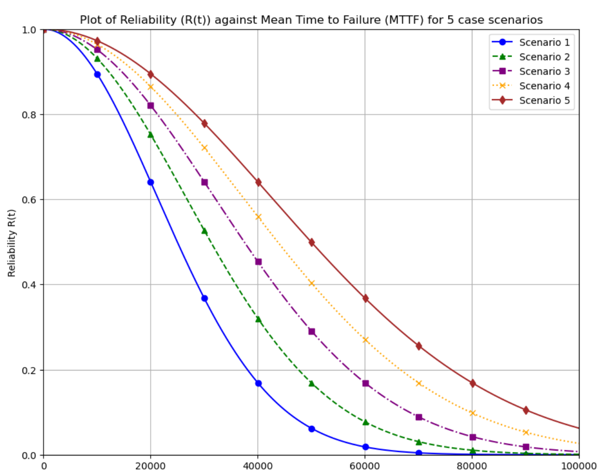
The steady-state probabilities for state $\mathcal{W} \geq 0.5$ indicate that the system has higher reliability and stability under the specified conditions. Hence, these data samples were extracted to conduct an additional sensitivity analysis that involved adjusting the individual component-level MEM and CPU consumption as well as varying the detection and migration rates which were earlier held constant. From the analysis, it was observed that the baseline scenario based on failure and repair rate data from [29], [33], [34] achieved a near-perfect availability of $\mathcal{A} = 0.99999$. However, adjustments in component failure rates and lowering MEM and CPU thresholds yielded availability scores between $93\% - 99.995\%$, slightly below the industry SLA of $\mathcal{A} = 99.999\%$ as illustrated in Figure 10(b). Initially, modifying repair rates from a standard 24-hour MTTR per component increased availability. Eventually, they hit a point of diminishing returns (optimal MTTF) where further improvement of the repair rates did not enhance system



(a) A violin plot showing the spread and central tendency of steady-state probabilities for different states in the HCTMM across 15 case scenarios (with varied failure and repair rates combinations).



(b) Varying individual component failure rates (λ_i) and increasing MEM and CPU thresholds



(c) Varying individual component repair rates (μ_i) and holding all other state transitions constant.

Fig. 10. The proposed Homogeneous Continuous Time Markov Model (HCTMM) steady-state behavior analysis with varying failure and repair rates.

reliability.

Furthermore, the system reliability was simulated over 100,000 hours ($\cong 11 years$) as illustrated in Figure 10(c) using the same scenarios. All the scenarios exhibited a system reliability $\gg 0.8$ within the first 30,000 hours ($\cong 3.425 years$). It is evident that within the first 3.5 years of operation, the monitoring agent successfully detects, recovers, and maintains the proposed network without human intervention. However, the reliability decreases with time and is implicitly attributed to stress acceleration in the offshore environment where external factors or stressors such as high temperature, excessive loads, or harsh environmental conditions lead to premature failures. Overall, scenario 5, demonstrates the best performance with the slowest decline in reliability, suggesting an effective implementation of the repair strategy while scenarios 1 and 2 depict the steepest decline representing the high failure rates associated with the two scenarios.

The results show that implementing a new detection and recovery strategy ensures consistent availability and reliability, meeting SLAs for the SDN-enabled IIoT-Edge network's suitability in the given application scenario. This study notes that the proposed network may also encounter service function chains (SFC) related failures next to the simulated failures of individual components. SFCs, which include network services like firewalls, load balancers, and intrusion detection systems arranged in a specific order, are managed and orchestrated by SDN controllers. These SDN controllers dynamically adjust the chain based on network conditions and policies, ensuring that data packets traverse the required services in the correct sequence for flexible and dynamic network management. Future studies may assess these SFC-related failures alongside the cost implications of the approaches taken to reduce failure rates and increase repair rates in the sensitivity analysis for the individual components in the control plane cluster and the data plane. In addition, future studies may assess the reliability and cost-effectiveness of scalable systems with increased redundancy factoring in the environmental failures in such offshore application scenarios.

VI. Conclusion

This paper assessed the dependability of SDN-enabled IIoT-Edge networks for next-generation offshore wind farm applications, proposing a new design that facilitates critical, latency-sensitive data transfer with a minimal number of redundant elements. Despite the advantages of using SDN to facilitate dynamic and centralized management of the offshore wind farm's IIoT-Edge network, stochastic failures such as random component malfunctions, software malfunctions, and CPU overconsumption, and memory leakages cause intermittent network service disruptions. The study introduced an innovative system model abstraction that mapped stochastic failures to both the control and data planes. It included a monitoring agent in the application plane that effectively detected these stochastic failures, facilitated recovery, and maintained the network in a fully operational state. To assess the dependability of the proposed SDN-enabled IIoT-Edge network, a two-pronged approach was used that (i) estimated key parameters and analyzed the transient behavior through a proof-of-concept simulation test bed and (ii) analyzed steady-state behavior using a probabilistic Homogeneous Continuous Time Markov model (HCTMM). The study demonstrated the impact of an effective strategy to detect, recover, and maintain the network by setting up scenarios with varied component failure and repair rates, starting from a baseline scenario. The steady-state availability of the system for the analyzed scenarios ranges between $\mathcal{A} = 93.0\%$ - 98% which is at or close to $\mathcal{A} = 99.999\%$ as stipulated in industry SLAs. Assessing the dependability of the proposed network guides the design of more effective network management strategies, especially for high availability and consistent performance-demanding environments.

ACKNOWLEDGEMENT

The authors would like to thank the Système Départment unit at EDF R&D (Paris-Saclay, France) for their substantial input in improving the architecture and system model, and for providing access to the Smart Grid laboratory resources.

REFERENCES

[1] L. Kou, Y. Li, F. Zhang, X. Gong, Y. Hu, Q. Yuan, and W. Ke, "Review on monitoring, operation and maintenance of smart offshore wind farms," *Sensors*, vol. 22, no. 8, p. 2822, 2022.

[2] A. Mwangi, K. Sundsgaard, J. A. L. Vilaplana, K. V. Vilerá, and G. Yang, "A system-based framework for optimal sensor placement in smart grids," in *2023 IEEE Belgrade PowerTech*, pp. 1–6, IEEE, 2023.

[3] G. Ertek and L. Kailas, "Analyzing a decade of wind turbine accident news with topic modeling," *Sustainability*, vol. 13, no. 22, p. 12757, 2021.

[4] F. Zhou, X. Tu, and Q. Wang, "Research on offshore wind power system based on internet of things technology," *International Journal of Low-Carbon Technologies*, vol. 17, pp. 645–650, 2022.

[5] C. A. Walford, "Wind turbine reliability: understanding and minimizing wind turbine operation and maintenance costs.," tech. rep., Sandia National Laboratories (SNL), Albuquerque, NM, and Livermore, CA . . . , 2006.

[6] G. Rinaldi, P. R. Thies, and L. Johanning, "Current status and future trends in the operation and maintenance of offshore wind turbines: A review," *Energies*, vol. 14, no. 9, p. 2484, 2021.

[7] Z. Ren, A. S. Verma, Y. Li, J. J. Teuwen, and Z. Jiang, "Offshore wind turbine operations and maintenance: A state-of-the-art review," *Renewable and Sustainable Energy Reviews*, vol. 144, p. 110886, 2021.

[8] R. Sikarwar, P. Yadav, and A. Dubey, "A survey on iot enabled cloud platforms," in *2020 IEEE 9th International Conference on Communication Systems and Network Technologies (CSNT)*, pp. 120–124, IEEE, 2020.

[9] X. Cao, Y. Xu, Z. Wu, X. Qin, and F. Ye, "Data acquisition and management of wind farm using edge computing," *International Journal of Grid and Utility Computing*, vol. 13, no. 2-3, pp. 249–255, 2022.

[10] P. Zhang, Z. He, C. Cui, C. Xu, and L. Ren, "An edge-computing framework for operational modal analysis of offshore wind-turbine tower," *Ocean Engineering*, vol. 287, p. 115720, 2023.

[11] F. Li, L. Li, and Y. Peng, "Research on digital twin and collaborative cloud and edge computing applied in operations and maintenance in wind turbines of wind power farm," in *Proceedings of the 2nd International Conference on Green Energy, Environment and Sustainable Development (GEESD2021)*, vol. 17, p. 80, IOS Press, 2021.

[12] H.-M. Chung, S. Maharjan, Y. Zhang, F. Eliassen, and T. Yuan, "Edge intelligence empowered uav s for automated wind farm monitoring in smart grids," in *GLOBECOM 2020-2020 IEEE Global Communications Conference*, pp. 1–6, IEEE, 2020.

[13] S. Siddiqui, S. Hameed, S. A. Shah, I. Ahmad, A. Aneiba, D. Draheim, and S. Dustdar, "Towards software-defined networking-based iot frameworks: A systematic literature review, taxonomy, open challenges and prospects," *IEEE Access*, 2022.

[14] N. Feamster, J. Rexford, and E. Zegura, "The road to sdn: an intellectual history of programmable networks," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 2, pp. 87–98, 2014.

[15] C. Urrea and D. Benítez, "Software-defined networking solutions, architecture and controllers for the industrial internet of things: A review," *Sensors*, vol. 21, no. 19, p. 6585, 2021.

[16] S. A. Gutiérrez, J. F. Botero, N. G. Gómez, L. A. Fletscher, and A. Leal, "Next-generation power substation communication networks: Iec 61850 meets programmable networks," *IEEE Power and Energy Magazine*, vol. 21, no. 5, pp. 58–67, 2023.

[17] A. K. Al Mhdawi and H. Al-Raweshidy, "μC-sdn: Micro cloud-software defined network testbed for onshore wind farm network recovery," in *2018 IEEE Global Conference on Internet of Things (GCIoT)*, pp. 1–6, 2018.

[18] P. Vizarreta, A. Van Bemten, E. Sakic, K. Abbasi, N. E. Petroulakis, W. Kellerer, and C. M. Machuca, "Incentives for a softwarization of wind park communication networks," *IEEE Communications Magazine*, vol. 57, no. 5, pp. 138–144, 2019.

[19] V. Netes, "New international standard for dependability," *Dependability*, vol. 3, pp. 54–58, 2016.

[20] V. Netes, "Dependability measures for access networks and their evaluation," in *2020 26th Conference of Open Innovations Association (FRUCT)*, pp. 352–358, IEEE, 2020.

[21] Z. Ma, M. Xiao, Y. Xiao, Z. Pang, H. V. Poor, and B. Vucetic, "High-reliability and low-latency wireless communication for internet of things: Challenges, fundamentals, and enabling technologies," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 7946–7970, 2019.

[22] W. Ahmad, O. Hasan, U. Pervez, and J. Qadir, "Reliability modeling and analysis of communication networks," *Journal of Network and Computer Applications*, vol. 78, pp. 191–215, 2017.

[23] P. E. Heegaard, B. E. Helvik, and V. B. Mendiratta, "Achieving dependability in software-defined networking—a perspective," in *2015 7th International Workshop on Reliable Networks Design and Modeling (RNDM)*, pp. 63–70, IEEE, 2015.

[24] L. Santos, B. Cunha, I. Fé, M. Vieira, and F. A. Silva, "Data processing on edge and cloud: a performability evaluation and sensitivity analysis," *Journal of Network and Systems Management*, vol. 29, no. 3, p. 27, 2021.

[25] L. De Simone, M. Di Mauro, M. Longo, R. Natella, and F. Postiglione, "Multi-provider ims infrastructure with controlled redundancy: A performability evaluation," *IEEE Transactions on Network and Service Management*, 2023.

[26] M. S. Jheeta, "Resilience, reliability, and recoverability (3rs)," Master's thesis, UiT The Arctic University of Norway, 2022.

[27] K. B. Misra, *Handbook of performability engineering*. Springer Science & Business Media, 2008.

[28] T. A. Nguyen, T. Eom, S. An, J. S. Park, J. B. Hong, and D. S. Kim, "Availability modeling and analysis for software defined networks," in *2015 IEEE 21st Pacific Rim International Symposium on Dependable Computing (PRDC)*, pp. 159–168, IEEE, 2015.

[29] F. Xiao, Z. Zhang, and X. Yin, "Reliability evaluation of the centralized substation protection system in smart substation: RELIABILITY EVALUATION OF CENTRALIZED SUBSTATION PROTECTION SYSTEM," *IEEJ Transactions on Electrical and Electronic Engineering*, vol. 12, pp. 317–327, May 2017.

[30] R. Kamal Kaur, B. Pandey, and L. K. Singh, "Dependability analysis of safety-critical systems: Issues and challenges," *Annals of nuclear energy*, vol. 120, pp. 127–154, 2018.

[31] F. Frattini, A. Bovenzi, J. Alonso, and K. Trivedi, "Reliability indices," *Wiley Encyclopedia of Operations Research and Management Science*, 2010.

[32] P.-A. Brameret, *Assessment of reliability indicators from automatically generated partial Markov chains*. PhD thesis, École normale supérieure de Cachan-ENS Cachan, 2015.

[33] P. Vizarreta, K. Trivedi, V. Mendiratta, W. Kellerer, and C. Mas-Machuca, "Dason: Dependability assessment framework for imperfect distributed sdn implementations," *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 652–667, 2020.

[34] V. B. Mendiratta, L. J. Jagadeesan, R. Hanmer, and M. R. Rahman, "How reliable is my software-defined network? models and failure impacts," in *2018 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, pp. 83–88, IEEE, 2018.

[35] G. Nencioni, B. E. Helvik, A. J. Gonzalez, P. E. Heegaard, and A. Kamisinski, "Availability modelling of software-defined backbone networks," in *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop (DSN-W)*, pp. 105–112, IEEE, 2016.

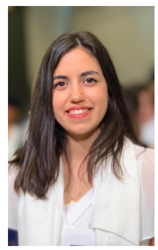
[36] F. Longo, S. Distefano, D. Bruneo, and M. Scarpa, "Dependability modeling of software-defined networking," *Computer Networks*, vol. 83, pp. 280–296, 2015.

[37] S. Moazzeni, M. R. Khayyambashi, N. Movahhedinia, and F. Callegati, "On reliability improvement of software-defined networks," *Computer Networks*, vol. 133, pp. 195–211, 2018.

[38] D. P. V. Kharchenko and A. P. M. Kolisnyk, "18. markov's modelling of iot systems," *Internet of Things for Industry and Human Applications*, p. 76.
[39] A. Mwangi, R. Sahay, E. Fumagalli, M. Gryning, and M. Gibescu, "Towards a software-defined industrial iot-edge network for next-generation offshore wind farms: State of the art, resilience, and self-x network and service management," *Energies*, vol. 17, no. 12, 2024.
[40] W. Zhang, J. Vatn, and A. Rasheed, "A review of failure prognostics for predictive maintenance of offshore wind turbines," in *Journal of Physics: Conference Series*, vol. 2362, p. 012043, IOP Publishing, 2022.
[41] E. Kaljic, A. Maric, P. Njemcevic, and M. Hadzialic, "A survey on data plane flexibility and programmability in software-defined networking," *IEEE Access*, vol. 7, pp. 47804–47840, 2019.
[42] N. Kabbara, A. Mwangi, M. Gibescu, A. Abedi, A. Stefanov, and P. Palensky, "Specifications of a simulation framework for virtualized intelligent electronic devices in smart grids covering networking and security requirements," in *2023 IEEE Belgrade PowerTech*, pp. 1–6, IEEE, 2023.
[43] E. Torres, P. Eguia, O. Abarrategi, D. Larruskain, V. Valverde, and G. Buigues, "Trends in centralized protection and control in digital substations," pp. 1–6, 2023.
[44] S. Hamadache, E. Benkhelifa, H. Kholidy, P. Kathiravelu, and B. B. Gupta, "Leveraging sdn for real world windfarm process automation architectures," in *2023 Tenth International Conference on Software Defined Systems (SDS)*, pp. 115–120, IEEE, 2023.
[45] A. Leal, M. Durán, and J. F. Botero, "Reliability provision in software defined power substations communication networks," *Computer Networks*, vol. 181, p. 107560, 2020.
[46] A. O. Jefia, S. I. Popoola, and A. A. Atayero, "Software-defined networking: current trends, challenges, and future directions," in *Proceedings of the International Conference on Industrial Engineering and Operations Management*, pp. 27–29, 2018.
[47] F. Liu, G. Kibalya, S. Santhosh Kumar, and P. Zhang, "Challenges of traditional networks and development of programmable networks," in *Software defined internet of everything*, pp. 37–61, Springer, 2021.
[48] M. Cabral, M. Silveira, and R. Urie, "Sdn advantages for ethernet-based control," *Schweitzer Engineering Laboratories, Inc., Tech. Rep*, 2019.
[49] K. Benzekki, A. El Fergougui, and A. Elbelrhiti Elalaoui, "Software-defined networking (sdn): a survey," *Security and communication networks*, vol. 9, no. 18, pp. 5803–5833, 2016.
[50] S. H. Yeganeh, A. Tootoonchian, and Y. Ganjali, "On scalability of software-defined networking," *IEEE Communications Magazine*, vol. 51, no. 2, pp. 136–141, 2013.
[51] B. Goswami, S. Hu, and Y. Feng, "Software-defined networking for real-time network systems," in *Handbook of Real-Time Computing*, pp. 935–959, Springer, 2022.
[52] R. Uddin, A. S. Alghamdi, M. H. Uddin, A. B. Awan, and S. A. Naseem, "Ethernet-based fault diagnosis and control in smart grid: a stochastic analysis via markovian model checking," *Journal of Electrical Engineering & Technology*, vol. 14, pp. 2289–2300, 2019.
[53] K. Choumas and T. Korakis, "When raft meets sdn: How to elect a leader over a network," in *2020 6th IEEE Conference on Network Softwarization (NetSoft)*, pp. 140–144, IEEE, 2020.
[54] J. Wu, Y. Huang, J. Kong, Q. Tang, and X. Huang, "A study on the dependability of software-defined networks," in *International Conference on Materials Engineering and Information Technology Applications (MEITA 2015)*, pp. 314–318, Atlantis Press, 2015.
[55] K. S. Trivedi and A. Bobbio, *Reliability and availability engineering: modeling, analysis, and applications*. Cambridge University Press, 2017.
[56] N. E. Petroulakis, G. Spanoudakis, and I. G. Askoxylakis, "Patterns for the design of secure and dependable software defined networks," *Computer Networks*, vol. 109, pp. 39–49, 2016.
[57] S. Secci, A. Diamanti, J. M. S. Vilchez, M. T. Bah, P. Vizzarreta, C. M. Machuca, S. Scott-Hayward, and D. Smith, *Security and performance comparison of ONOS and ODL controllers*. PhD thesis, ONOS, 2019.
[58] L. Zhu, M. M. Karim, K. Sharif, C. Xu, F. Li, X. Du, and M. Guizani, "Sdn controllers: A comprehensive analysis and performance evaluation study," *ACM Computing Surveys (CSUR)*, vol. 53, no. 6, pp. 1–40, 2020.
[59] I. Kabashkin, "Availability of services in wireless sensor network with aerial base station placement," *Journal of Sensor and Actuator Networks*, vol. 12, no. 3, p. 39, 2023.
[60] B. Luksik, A. Loveless, and A. D. George, "Gatekeeper: A reliable reconfiguration protocol for real-time ethernet systems," in *2021 IEEE/AIAA 40th Digital Avionics Systems Conference (DASC)*, pp. 1–10, IEEE, 2021.
[61] S. Becker, T. Pfandzelter, N. Japke, D. Bermbach, and O. Kao, "Network emulation in large-scale virtual edge testbeds: A note of caution and the way forward," in *2022 IEEE International Conference on Cloud Engineering (IC2E)*, pp. 1–7, IEEE, 2022.

Biography Section

Agrippina Mwangi Received her BSc. in Electrical and Electronic Engineering from Dedan Kimathi University of Technology and MSc. in Electrical and Computer Engineering from Carnegie Mellon University. She is pursuing a Ph.D. in the Energy & Resources Group at Copernicus Institute of Sustainable Development, Utrecht University (The Netherlands). Her research interests are IoT Edge, SDN, NFV, autonomous networks, and offshore wind.



Nadine Kabbara Received her MSc. degree in control and electrical engineering from Paris Saclay University. She is an Industrial Ph.D. working with the French utility EDF in their R&D department and Utrecht University (The Netherlands). She is also part of the InnoCyPES European project on connected digital smart grids. Her research interests include digital substations, IEC 61850, virtualized IEDs, data modeling, and simulation frameworks.

Patrick Coudray is a research engineer at EDF Lab in Système Départment that deals with Smart Grids. He is an expert in the field of telecommunications for smart grids: IoT, 5G, and beyond. He also works on the subject of cross-resilience of Telecom/electricity networks.

Mikkel Gryning Received his M.Sc. and Ph.D. in Electrical Engineering from the Technical University of Denmark. Working as Senior Lead Specialist at Ørsted in SCADA and Communication. Research interests include power system stability robustness evaluation, hybrid power plant integration and design, and communication systems for offshore wind power plants.

Madeleine Gibescu is a full professor in the area of Integration of Intermittent Renewable Energy in the Copernicus Institute of Sustainable Development, Faculty of Geosciences, Utrecht University. Dr. Gibescu received her Ph.D. in Electrical Engineering from the University of Washington, Seattle, Washington, U.S. in 2003. Her research interests are in the modeling and simulation of power systems and electricity markets with a large penetration of renewable energy sources. Since 2004, Prof. Gibescu has worked in the Netherlands in various academic capacities at TU Delft, TU Eindhoven, and Utrecht University, leading research in smart grids and system integration of wind energy and solar photovoltaics.