Assuring Autonomy of UAVs in Mission-critical Scenarios by Performability Modeling and Analysis

ERMESON ANDRADE, Federal Rural University of Pernambuco, Brazil, ORCID: 0000-0002-9614-4492
FUMIO MACHIDA, University of Tsukuba, Japan, ORCID: 0000-0001-8359-8535

Uncrewed Aerial Vehicles (UAVs) have been used in mission-critical scenarios such as Search and Rescue (SAR) missions. In such a mission-critical scenario, flight autonomy is a key performance metric that quantifies how long the UAV can continue the flight with a given battery charge. In a UAV running multiple software applications, flight autonomy can also be impacted by faulty application processes that excessively consume energy. In this paper, we propose FA-Assure (Fight Autonomy assurance) as a framework to assure the autonomy of a UAV considering faulty application processes through performability modeling and analysis. The framework employs hierarchically-configured stochastic Petri nets (SPNs), evaluates the performability-related metrics, and guides the design of mitigation strategies to improve autonomy. We consider a SAR mission as a case study and evaluate the feasibility of the framework through extensive numerical experiments. The numerical results quantitatively show how autonomy is enhanced by offloading and restarting faulty application processes.

CCS Concepts: • **Computer systems organization → Dependable and fault-tolerant systems and networks**.

Additional Key Words and Phrases: Autonomy, Hierarchical model, Search and rescue, Stochastic Petri nets, UAV

ACM Reference Format:
Ermeson Andrade and Fumio Machida. 2023. Assuring Autonomy of UAVs in Mission-critical Scenarios by Performability Modeling and Analysis. 1, 1 (August 2023), 25 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

The unmanned aerial vehicles (UAVs) market, commonly known as drones, is relatively new but is growing quickly. UAVs have been used for various purposes in real-world, especially in mission-critical situations [33] like natural disasters [10], public safety [22] and remote workplaces [5]. For remote monitoring and surveillance operations, UAVs can collect data faster and more accurately than the traditional approach using satellite imagery. Compared with manned aircraft, UAVs are more flexible and cheaper to acquire, maintain, and operate [28]. More importantly, they can deal with potentially dangerous situations without human attendance.

As UAVs are flexibly designed and simpler to deploy, they were initially used mainly by the military in environments that allow a relatively large margin for error. However, with the increased use of UAVs within the national airspace, they have been required to operate at high availability levels that approach or exceed manned aircraft operations [8]. As stated in [27], the failure of UAVs can often cause effects that range from inconvenience and irritation to severe impact on society and on its environment. Therefore, reliability testing and assurance of UAVs are becoming a critical challenge in the design, development, and operation of UAV-based systems. The flight controller is the most critical software

Authors' addresses: Ermeson Andrade, Federal Rural University of Pernambuco, Recife, Brazil, ORCID: 0000-0002-9614-4492, ermeson.andrade@ufrpe.br; Fumio Machida, University of Tsukuba, Tsukuba, Japan, ORCID: 0000-0001-8359-8535, machida@cs.tsukuba.ac.jp.

in UAVs that receives sensor input data and navigates the device [9]. Recent studies discuss the reliability issues of flight controllers since it directly impacts mission performance and safety [14, 16, 34, 36]. However, the reliability of less critical software components, for example, application programs that may not directly impact the UAV control, has been paid little attention. UAVs may have installed multiple application programs that can be vulnerable and subject to frequent updates. It is a practical issue that a faulty application can lead to wrong or undesirable behavior, indirectly threatening the UAV flight autonomy.

To address the issue, this study aims to analyze the impact of faulty application programs used in UAV systems on their mission performance. As a UAV is a battery-powered device, flight autonomy is always constrained by the battery charge. Evaluating autonomy-related metrics like the probability of running out of battery is essential for mission-critical scenarios because they can measure how long a UAV can survive in the mission period. Similar to other battery-powered devices, energy bugs [15] are a real concern for continuous system operation as they may induce energy over-consumption and rapid battery discharge. Application behaviors in association with resource and energy consumption need to be carefully investigated in the design and evaluation of UAV systems.

In this paper, we propose an autonomy assurance framework called *FA-Assure* for UAV systems in a mission-critical scenario that is at risk of faulty applications impacting flight autonomy. We consider a high-workload process and a software aging process as examples of faulty processes that potentially reduce the autonomy of a UAV indirectly by excessively consuming resources. FA-Assure employs hierarchically-configured stochastic Petri nets to capture the dynamics of application processes with recovery methods and to evaluate the performability-related metrics for given environmental conditions. Quantitative performability evaluation allows us to derive the appropriate operation strategy to mitigate faulty applications' impact and maintain good performance and availability. Considering a SAR mission, we conduct extensive numerical studies to show how the performability-related metric is evaluated using the hierarchical models. As a SAR-based UAV is assisted by real-time image processing, the image processing application is likely the heaviest process running on the UAV. Using fog computing infrastructure to accept computation offloading of image processing tasks is a viable solution to improve the UAV performance and availability [17, 18]. While the quality of the wireless network is critical for computation offloading, we show how the effectiveness of offloading is significantly improved by redundant communication channels. Moreover, we derive a strategy to choose the computation offloading mode and to restart faulty processes to meet the mission requirements through sensitivity analysis.

To summarize, the contributions of the work are as follows.

- The study addresses the risk of application programs running on a UAV that may impact flight autonomy and mission completion performance.
- To facilitate the reliability design of a UAV system for mission-critical scenarios, FA-Assure is proposed. We provide the models to capture the dynamic behaviors of application programs on the UAV under uncertain environments.
- A case study of a search and rescue mission is presented to show the feasibility and effectiveness of the proposed framework. Extensive numerical studies are conducted to analyze the impact of several internal and external factors on autonomy-related metrics.

We structure the remainder of the paper as follows. Section 2 describes the related work. Section 3 explains the potential risk of faulty applications on a UAV system. Section 4 introduces FA-Assure: the proposed autonomy assurance framework. Section 5 details the SPN models used in FA-Assure for evaluating the autonomy-related metrics. Section

6 shows the case study and the results of numerical experiments. Lastly, Section 7 concludes this work and briefly discusses future works.

2 RELATED WORK

UAVs are a relatively new technology that has attracted the attention of researchers. The investigations around this concept are broad and cover several areas. In [31], a detailed survey on the different aspects of UAV environments is presented.

In the last few years, many studies have been proposed for the modeling and evaluation of UAV environments. In [18, 19], the authors proposed using stochastic reward nets (SRNs) to analyze performance-availability trade-offs in image processing tasks running on drones and fog nodes. In [17], the authors proposed stochastic models to estimate the availability, performance, and energy consumption of a drone-based image processing system considering different computation modes, such as single drone processing, fog offloading, and drone load-balancing.

In [40], the authors presented a review of energy consumption models for drone delivery operations. They identified the main factors that affect the energy consumption of drones and discussed the similarities and differences between various models. Nguyen *et. al* [25] proposed to adopt fault trees and stochastic Petri nets to model and analyze the dependability of a typical unmanned aerial system. The authors also performed sensitivity analysis on the SPN models to investigate the elements that most impact on the system's overall availability and reliability.

In a mission-critical context, a few studies have considered UAV-based environments. In [30], drone technology was used for the inspection of offshore wind infrastructures in order to improve operational safety and minimize production downtime. The authors stated that the use of drones for such type of inspection is still in an early stage of development and its suitability has not yet been proven. Thus, they proposed a semi-quantitative reliability analysis framework to determine the criticality of failures that can occur in inspection missions using drones. In [32], the authors developed and tested a drone-based environment for avalanche rescue activities based on an avalanche transceiver. The results showed that the proposed environment is capable of locating survivors quickly and, therefore, guaranteeing the maximum chance of survival. For assuring the safety of UAV systems, a tool for testing the safe behavior of UAVs with a scenario-based testing approach is presented [29].

While our study also considers a UAV system in a mission-critical scenario, we first look into the risk of faulty application processes that is different from the studies concerning the issues of the flight controller [14, 16, 34, 36]. We demonstrate how continuous application programs running on the UAV may adversely impact mission performance by quantitatively analyzing UAV's flight autonomy. Regarding the modeling approach, we use SPNs like existing studies [17, 18, 25], but we present a hierarchically-configured model to compute the autonomy-related metrics. Since the autonomy-related metrics depend on the conditions of different components, such as application processes and wireless communication channels, unlike other studies, SPN subnets need to be configured dependently. The constructed hierarchical model allows us to evaluate flight autonomy and guide a desirable configuration that meets the mission requirements.

Software faults impacting the operation of battery-powered devices have been studied in the literature [7, 15, 26, 35]. The authors of [15] proposed methods to detect two types of energy bugs in Android applications. The tool for verifying the absence of no-sleep energy bugs is presented in [35]. Several studies on energy bugs in smartphone platforms are summarized in [26]. UAVs can suffer from such energy bugs as they are also battery-powered devices. For a mission-critical scenario, it is crucial to ensure the autonomy that is constrained by the battery charge. Our case study involves a UAV that must complete its mission without needing to charge its battery and must also deal with potentially unstable network connectivity over the course of its operation. In contrast to other systems with battery-driven devices, these unique characteristics cannot be guaranteed. The design and operation of our UAV must account for these factors to ensure successful mission completion.

3 MOTIVATION

Application programs are often subject to changes and updates, potentially becoming a vulnerable component in an integrated software system. Recent studies for software reliability of drone systems mainly focus on flight control software such as PX4 and Ardupilot as they have a direct impact on the safe operation of UAVs [34]. However, the behavior of application processes is not negligible since it may harm the UAV mission indirectly. For instance, a compute-intensive program with a high demand may consume a considerable amount of CPU and energy, which must have adverse impacts on the mission performance. Since battery charge is a critical constraint of UAV systems, application programs need to be energy-efficient.

To understand the potential risk of faulty application processes on an edge computing device, we conducted experiments using a small testbed. Considering a UAV-based SAR mission, we ran a real-time object detection algorithm on a Raspberry Pi 4 with a 1.5GHz quad-core ARMv8 CPU and 4GB RAM. We did not use a UAV for this experiment since the experiments may damage the property if it becomes out of control. We assume the UAV application needs to be executed in a restricted computation environment like the edge device we used.

Our first experiment aims to characterize the impact of workload by the object detection algorithm on battery usage. For this purpose, we use a UPS Lithium battery power module with 5000 mAh to empower the Raspberry Pi instead of connecting it to a power outlet. Following the configuration used in [37], we utilized an open-source implementation of YOLOv5 based on the PyTorch framework and manipulated the workload by configuring different input image sizes: 1280x720 and 640x360. For each workload, we ran YOLOv5 for one hour and recorded both the frame per second and battery usage. The results are presented in Table 1. The frame per second for the process with 640x360 is higher than the case with 1280x720, resulting in a higher workload. The impact of the workload on the battery usage is apparent; the object detection algorithm under a higher workload condition (640x360) consumes more energy (78%). The rapid battery consumption due to the heavy-loaded application can be a risk to the autonomy of battery-powered devices.

Table 1. Battery consumption in one hour of real-time object detection with different workloads.

Input image size	Frame per second	Battery consumption
640x360	0.7319	78%
1280x720	0.2999	69%
No	N/A	40%

Next, we conducted another experiment to investigate the reliability issue of an application program due to excessive resource consumption over time. We used the same configuration described in Watanabe *et. al* [37] and ensured a continuous execution over an extended period of time by connecting the device to a power outlet. The YOLOv5 is run with a 640px image size of a video stream. Figure 1 shows the observed trends of the free memory and swap usage over 200 hours of execution. As can be seen, the amount of free memory decreased sharply in the first 15 hours as represented by the dotted line. While the system is working even after that phase, the swapping eventually starts around 190 hours (plotted by the solid line). This increased swapping leads to significant I/O and CPU usage, which can adversely affect the object detection process and drain the battery of UAVs. Although a UAV mission is not supposed to

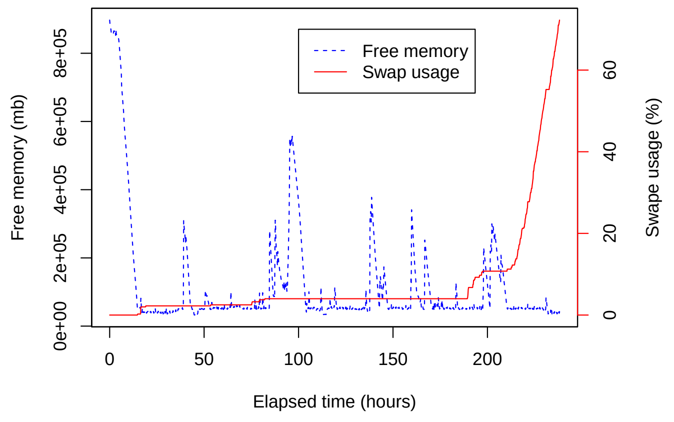


Fig. 1. Software aging in an object recognition application.

have such a long run, the results show the potential risk of software aging in the object detection process on UAVs. It is worth emphasizing that we did not collect energy consumption data for this specific experiment, as it was conducted on a device connected to a power source. Nevertheless, it is well-documented that memory leaks can indeed impact battery-powered devices, such as smartphones, as discussed in [38].

4 AUTONOMY ASSURANCE FRAMEWORK

To address the issue of a faulty application program in a mission-critical UAV environment, we propose FA-Assure as a framework to assure the autonomy of a UAV through performability modeling and analysis. First, we explain the fault models and the corresponding recovery methods considered in the framework. Then, the architecture of FA-Assure is explained.

4.1 Fault models

As explained in the motivation, we focus on the impact of a faulty application program such as an object detection algorithm on the mission of a UAV. We consider two types of software fault models that may have an impact on flight autonomy.

- **Heavy-loaded process**: A software process operating under a high workload can significantly consume CPU resources. While the application continues to function as intended, heavily-loaded applications can potentially disrupt the entire system by causing resource conflicts, among other issues. Moreover, increased CPU usage accelerates battery consumption, a critical factor for UAV flight.
- **Software aging process**: A software process can progressively consume a substantial amount of memory during continuous execution due to aging-related bugs [11]. These bugs have been observed in various application programs, such as image classifiers and object detection algorithms [2]. Software aging processes can also trigger

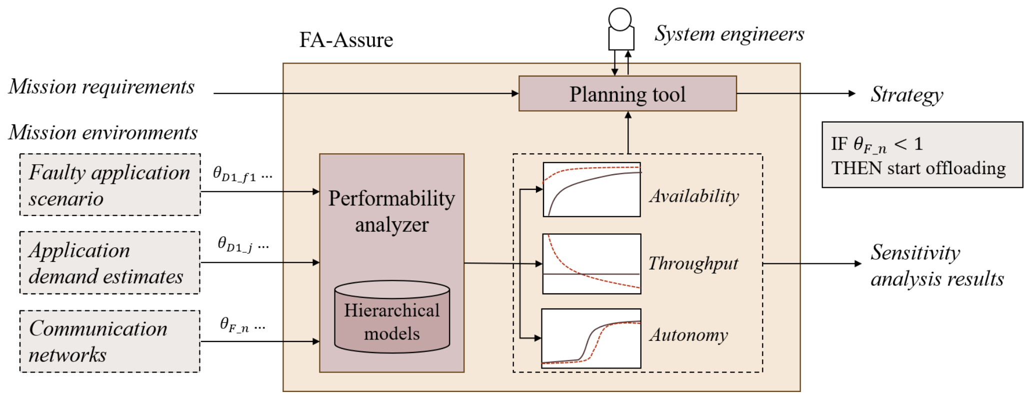


Fig. 2. Overview of FA-Assure.

swapping when the system's available memory is insufficient. Extensive swapping can result in elevated CPU usage, potentially affecting battery life.

4.2 Recovery methods

To counteract the application program faults in operation, we consider two recovery methods. For the heavy-loaded process, we can offload the process to another computing node to save the battery on the drone. Regardless of faulty and non-faulty processes, CPU-intensive applications reduce the battery charge significantly, and hence offloading can offer an effective mitigation. In this paper, we consider that a fog computing infrastructure is available for a UAV to allocate necessary computing resources for taking over the computation tasks. A fog computing infrastructure is a decentralized computing architecture that brings computation and data storage closer to the edge of the network [4]. Note that computation offloading is effective only when the UAV has a stable wireless network connection.

For a software aging process, the aging state can be cleared by process restart upon the aging detection. It requires a short downtime of the application process, but the process can be restarted with an initial robust state. Since other benign internal states in memory are also cleared, the restart needs to be carefully scheduled such that the application does not lose critical data in processing. Offloading can also mitigate the issue of software aging if the assigned fog node has a sufficient amount of memory.

4.3 FA-Assure

To analyze the impact of application faults and the effectiveness of the recovery methods for a UAV system, we propose FA-Assure which is an autonomy assurance framework that consists of the performability analyzer and the planning tool. Figure 2 shows the schematic overview of FA-Assure.

FA-Assure necessitates several input parameters about the internal and external factors of the UAV environment. The internal factors include the states of the application process running on the system, which may be idle, processing, aging, or failed. On the other hand, the external factors include application demands and the communication network states. The performability analysis is conducted by a performability analyzer in which hierarchically-configured SPNs are used to evaluate several performance measures such as availability, throughput, and autonomy. The estimated performance and the mission requirements, such as the mission period, are considered together for making the strategy that determines the computation mode and recovery operation to satisfy the mission requirements. The planning tool assists system designers to derive such a strategy from the results of sensitivity analysis on the hierarchical models. The strategy specifies the recommended computation mode and recovery operations for given system conditions. For example, the output strategy may suggest computation offloading of the application process if the communication delay of the fog node θ_{F_n} is less than 1 second. As the strategy-making process involves manual operations and interactions with engineers, an interface to access the analysis results is necessary. In this work, we assume the usage of software packages like SPNP [6] and Mercury [20] that provide graphical user interfaces as well. In this paper, we focus on the core part of FA-Assure, which is the performability models detailed in the next section.

5 MODELS

In this section, we discuss the models for assessing the performability of a UAV system under different mission environments. First, we introduce the formalism adopted in this work. After that, we detail the models for heavy-loaded processes (drone processing (DP) and fog processing (FP) modes). In DP, faulty application processes, which we assumed to be an image processing program such as an object detector, are continuously executed on the drone. In FP, on the other hand, captured images are offloaded to a fog node in a fog computing infrastructure instead of being processed on the drone. Next, we present the models for aging drones (AD), where aging states and a restart mechanism are considered. Finally, we explain the evaluated metrics: availability, throughput, drone flight autonomy, and out-of-battery probability. In the proposed models, we assume that all inter-event times are exponentially distributed in our analysis. Since we focus on the early phase of the UAV system design, detailed distributions may not be available. However, we can also assign general distributions to the models once the information is given or collected.

5.1 Stochastic Petri nets

Petri nets are a family of formalisms suitable for modeling various types of systems due to their capacity to represent concurrency, synchronization, communication mechanisms, and both deterministic and probabilistic delays. However, the original Petri net lacks the concept of time when analyzing performance and dependability. To address this, timed Petri nets were developed, incorporating event durations. In this work, we employ the stochastic Petri net [24], a specific type of timed Petri net, where activity delays (represented by transitions) are modeled as random variables with an exponential distribution. Marsan et al. [1] proposed the Generalized Stochastic Petri Net (GSPN) as an extension of SPN, which considers two types of transitions: timed and immediate. Timed transitions have exponentially distributed firing times, while immediate transitions, by definition, fire instantly without any delay. For conciseness, we use the acronym SPN to refer to the entire family of models derived from the original stochastic Petri net defined in [24].

5.2 DP and FP models

Figure 3 shows the SPNs that represent DP mode, while Table 2 presents the legend for these models. It is composed of three submodels corresponding to a drone, multiple network interfaces, and a battery. The drone model is shown in Figure 3a. It represents the states of an image processing on the drone. When a token is deposited in P_{D1_i}, representing that the drone has no image for processing, the transitions T_{D1_j} and T_{D1_f2} are enabled. When T_{D1_j} fires, a token is consumed from P_{D1_i} and a new token is deposited in P_{D1_p}. This state transition corresponds to the arrival of an image processing request. We assume that the arrival of image processing requests follows a Poisson process. A higher arrival rate causes the increased probability of the busy state leading to a heavy-workload condition. The firing of T_{D1_s} represents the completion of an image processing. On the other hand, when T_{D1_f2} fires, which represents an

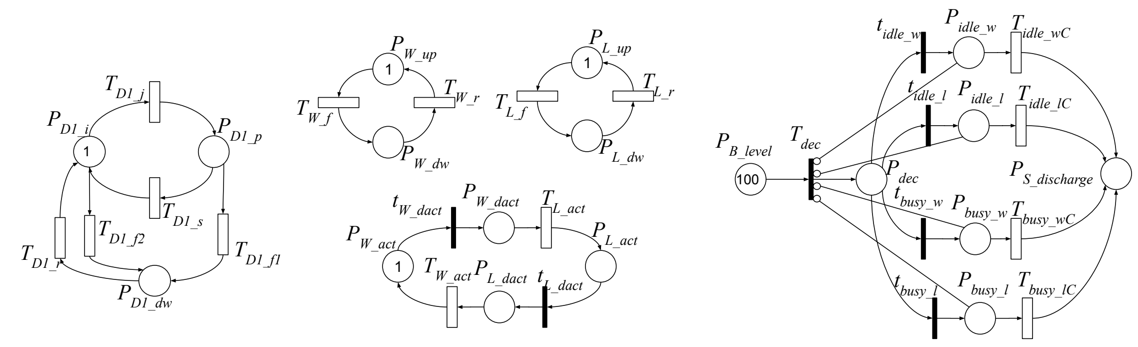


Fig. 3. Drone processing models.

application drone failure event in the idle state, a new token is deposited in P_{D1_dw}. The place P_{D1_dw} represents the application failure. The drone application can also fail while processing an image by firing T_{D1_f1}. When the drone application fails, the application undergoes the unavailable state and remains in this state until T_{D1_r} is fired, which represents the repair of the application. Once T_{D1_r} is fired, a token is consumed from P_{D1_dw} and a new one is added to P_{D1_i}.

Although there are many network communication technologies used by drone-based environments [12], in this work, the environment's network is modeled considering Wi-Fi and LTE connections (see Figure 3b) [23]. They represent the connectivity between the drone and the ground station so that the drones are only able to connect to the ground station if there is an active Wi-Fi or LTE connection. The models of the upper part of Figure 3b represent the connection and disconnection behavior of the Wi-Fi and LTE networks, respectively. The other model (see the bottom part of Figure 3b) represents the activation of either Wi-Fi or LTE. A token in the place P_{W_act} indicates that the Wi-Fi is activated. If the Wi-Fi goes down (represented by a token in P_{W_dw}), the LTE is activated. When the Wi-Fi connection is restored, the LTE is deactivated, for the sake of performance and energy consumption. Such behavior is modeled by guard functions assigned to the immediate transitions t_{W_dact} and t_{L_dac}, as described in Table 3.

Figure 3c presents the SPN for the battery discharge process. The battery consumption of a UAV device can be affected mainly due to two factors: network communication technologies and drone processing states. Consequently, the battery discharge process can be modeled using four immediate transitions t_{idle_w}, t_{idle_l}, t_{busy_w} and t_{busy_l}. The transitions t_{idle_w} and t_{idle_l} are associated with the probabilities of battery discharge when the drone is in the idle state and using Wi-Fi and LTE, respectively. On the other hand, the transitions t_{busy_w} and t_{busy_l} are associated with the probability of battery discharge when the drone is in a busy state and using Wi-Fi and LTE network connections, respectively. The place P_{B_level} contains one hundred tokens representing the battery is fully charged. By removing tokens from this place, the model represents the process of discharging the drone's battery. The discharge is modeled considering a fixed consumption of 1%. When T_{dec} fires, which means the beginning of the battery consumption, a new token is deposited in P_{dec}. After that, any of the transitions t_{idle_w}, t_{idle_l}, t_{busy_l} and t_{busy_l} can fire based on the probability assigned to these transitions. Note that these probabilities are obtained from the models in Figures 3a and b. Supposing t_{idle_w} fires, a token is consumed from P_{dec} and a new token is deposited in P_{idle_w}. This state transition means that 1% of the drone's battery is going to be consumed. The firing of T_{idle_wC} represents the completion of 1%

Table 2. Legend for the drone processing models.

Model	**Transition**	**Description**
Drone	P_{D1_i}	Drone idle state
	P_{D1_p}	Drone busy state
	P_{D1_dw}	Drone down state
	T_{D1_j}	Drone job arrival time
	T_{D1_s}	Drone service time
	T_{D1_f1}	Drone failure time in busy state
	T_{D1_f2}	Drone failure time in idle state
	T_{D1_r}	Drone recovery time
Network	P_{W_up}	Wi-Fi up state
	P_{W_dw}	Wi-Fi down state
	T_{W_f}	Wi-Fi disconnection time
	T_{W_r}	Wi-Fi connection time
	P_{L_up}	LTE up state
	P_{L_dw}	LTE down state
	T_{L_f}	LTE disconnection time
	T_{L_r}	LTE connection time
	P_{W_act}	Wi-Fi activation state
	P_{W_dact}	Wi-Fi deactivation state
	P_{L_act}	LTE activation state
	P_{L_dact}	LTE deactivation state
	t_{W_dact}	Wi-Fi deactivation
	t_{L_dact}	LTE deactivation
	T_{L_act}	LTE activation time
	T_{W_act}	Wi-Fi activation time
Battery	P_{B_level}	Battery level state
	$P_{S\_discharge}$	Battery discharge state
	T_{dec}	Decision point
	P_{dec}	Decision state
	t_{idle_w}	Drone probability (Wi-Fi and idle)
	t_{idle_l}	Drone probability (LTE and idle)
	t_{busy_w}	Drone probability (Wi-Fi and busy)
	t_{busy_l}	Drone probability (LTE and idle)
	P_{idle_w}	Battery discharge (Wi-Fi and idle) state
	P_{idle_l}	Battery discharge (LTE and idle) state
	P_{busy_w}	Battery discharge (Wi-Fi and busy) state
	P_{busy_l}	Battery discharge (LTE and busy) state
	T_{idle_wC}	Battery discharge (Wi-Fi and idle) time
	T_{idle_lC}	Battery discharge (LTE and idle) time
	T_{busy_wC}	Battery discharge (Wi-Fi and busy) time
	T_{busy_lC}	Battery discharge (LTE and busy) time

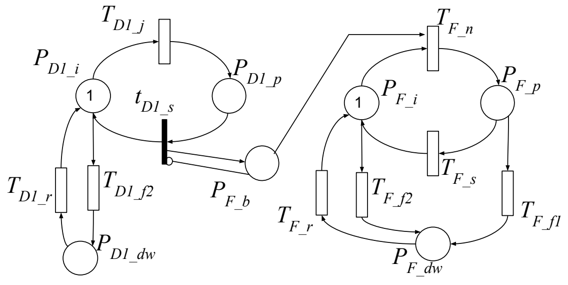
drain on the drone's battery. Lastly, tokens in $P_{S\_discharge}$ represent the battery's state of charge where one hundred tokens in this place mean the battery is fully discharged.

Figure 4 presents the models for FP mode, while Table 4 presents the legend for these models. It is composed of three submodels corresponding to a drone-fog environment, multiple network interfaces, and a battery. While the network

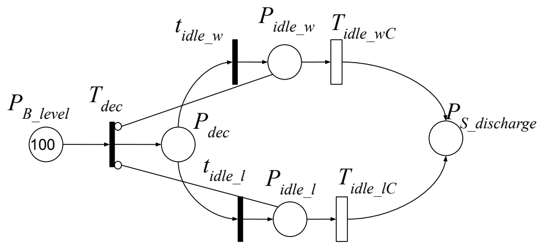
Table 3. Guard functions for the models of Figure 3.

Transition	Expression
t_{W_dact}	$(P_{W\_dw} = 1)$ AND $(P_{L\_up} = 1)$
t_{L_dact}	$(P_{W\_up} = 1)$

model is unchanged (we omitted from Figure 4), the drone-fog environment and battery models are changed from the drone processing models shown previously.



(a) Drone-Fog environment model



(b) Battery model

Fig. 4. Fog processing models.

In the fog offloading mode, the captured images are not processed on the drone, but sent to the fog node through the network. Therefore, the drone model has the place P_{F_b}, representing the drone offloading. Since the offloading only works when the WI-FI or LTE communication links are available, such condition is specified by the guard function assigned to t_{D1_s} (see Table 5). When a token is deposited in P_{F_b}, the transition T_{F_n}, which represents the network delay, is enabled. When T_{F_n} fires, tokens in P_{F_b} and P_{F_i} are removed and a new token is deposited in P_{F_p}. This state transition corresponds to the arrival of an image processing request in the fog. The firing of T_{F_s} represents the completion of an image processing. As the failure and recovery processes are the same as in the previous models presented, we do not detail them. Additionally, the battery model (see Figure 4b) is slightly changed because the captured images are not processed on the drone. Consequently, the part of the model related to the drone's busy state (e.g.: transitions t_{busy_w} and t_{busy_l}) is removed. The battery model hierarchically depends on the drone-fog environment and network models, since battery discharge rates should change according to processing and communications states.

5.3 AD models

Figure 5 presents the model we created based on the model of Figure 3 to consider aging states and a restart mechanism for a drone application. Table 6 presents the legend for the SPN models in Figure 5. We opted not to consider the models presented in Figure 4 as the processing occurs within a fog node. In this context, periodic restarts of the drone application, triggered promptly upon detecting aging, would have no impact on this environment. Although the network and battery models remain unchanged (omitted from Figure 5), we introduced a restart model (see Figure 5b) and updated the drone model to account for aging states (P_{AD1_i} and P_{AD1_p}) and a restart mechanism (T_{R1} and T_{R2}). The restart model represents the restarting of the application process by the firing of the transitions T_{R1} or T_{R2} when an aging state is detected, represented by the presence of tokens in the places P_{AD1_i} or P_{AD1_p}. The application can fail

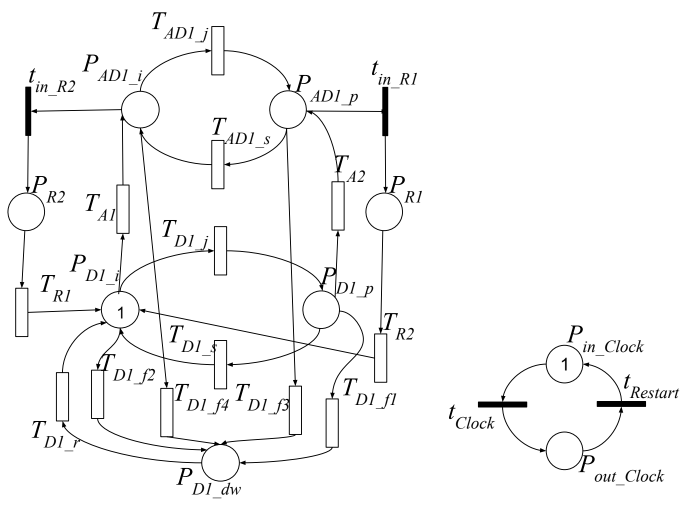
Table 4. Legend for the fog processing models.

Models	Transition	Description
Drone-Fog	P_{D1_i}	Drone idle state
	P_{D1_p}	Drone busy state
	P_{D1_dw}	Drone down state
	T_{D1_j}	Drone job arrival time
	T_{D1_f2}	Drone failure time in idle state
	T_{D1_r}	Drone recovery time
	t_{D1_s}	Drone offloading
	P_{F_b}	Drone offloading state
	P_{F_i}	Fog idle state
	P_{F_p}	Fog busy state
	P_{F_dw}	Fog down state
	T_{F_n}	Fog network delay
	T_{F_s}	Fog service time
	T_{F_f1}	Fog failure time in busy state
	T_{F_f2}	Fog failure time in idle state
	T_{F_r}	Fog recovery time
Battery	P_{B_level}	Battery level state
	$P_{S\_discharge}$	Battery discharge state
	T_{dec}	Decision point
	P_{dec}	Decision state
	t_{idle_w}	Drone probability (Wi-Fi and idle)
	t_{idle_l}	Drone probability (LTE and idle)
	P_{idle_w}	Battery discharge (Wi-Fi and idle) state
	P_{idle_l}	Battery discharge (LTE and idle) state
	T_{idle_wC}	Battery discharge (Wi-Fi and idle) time
	T_{idle_lC}	Battery discharge (LTE and idle) time

Table 5. Additional guard functions adopted for the models of Figure 4.

Transition	Expression
t_{D1_s}	$((P_{W\_up} = 1)$ AND $(P_{W\_act} = 1))$ OR $((P_{L\_up} = 1)$ AND $(P_{L\_act} = 1))$

from the aging states through the transitions T_{D1_f3} and T_{D1_f4}. Table 7 lists the guard functions for the model. When the application process is restarted, the state is changed to the normal state, as indicated by the guard functions of the transitions t_{in_R1}, t_{in_R2}, t_{Clock}, and $t_{Restart}$.



(a) Drone model (b) Restart model

Fig. 5. Aging drone processing models with a restart.

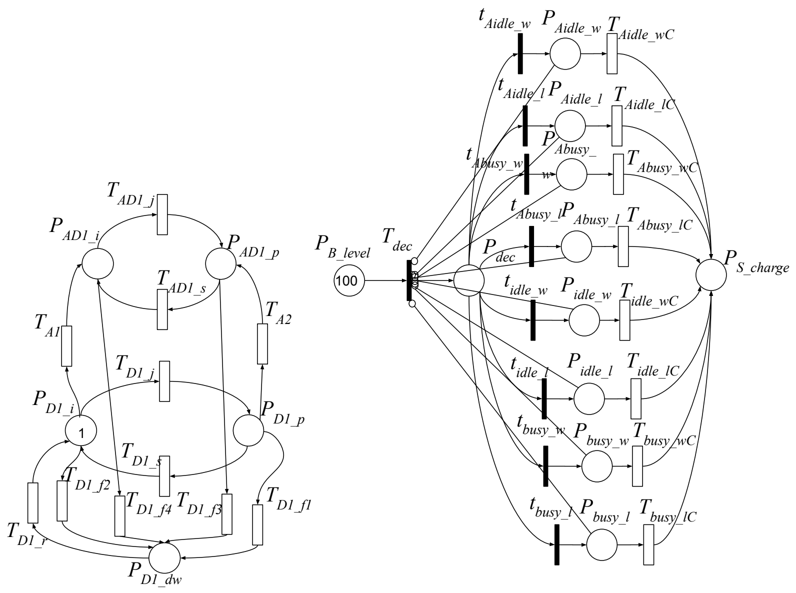
Table 6. Legend for the drone processing models with a restart.

Model	Transition	Description
Drone	P_{AD1_i}	Drone aging idle state
	P_{AD1_p}	Drone aging busy state
	T_{AD1_i}	Drone aging job arrival time
	T_{AD1_p}	Drone aging service time
	T_{A1}	Drone aging time in idle state
	T_{A2}	Drone aging time in busy state
	T_{D1_f3}	Drone failure time in aging busy state
	T_{D1_f4}	Drone failure time in aging idle state
	T_{in_R1}	Drone restart in busy state
	T_{in_R2}	Drone restart in idle state
	P_{R1}	Drone restart state (busy)
	P_{R2}	Drone restart state (idle)
	T_{R1}	Drone restart time (busy)
	T_{R2}	Drone restart time (idle)
	P_{D1_i}	Drone idle state
	P_{D1_p}	Drone busy state
	P_{D1_dw}	Drone down state
	T_{D1_j}	Drone job arrival time
	T_{D1_s}	Drone service time
	T_{D1_f1}	Drone failure time in busy state
	T_{D1_f2}	Drone failure time in idle state
	T_{D1_r}	Drone recovery time
Clock	P_{in_clock}	Drone aging detection state
	P_{out_clock}	Drone restart state
	t_{clock}	Drone aging detection
	$t_{restart}$	Drone restart

Table 7. Guard functions for the models of Figure 5.

Transition	Expression
t_{in_R1}	$(P_{in\_Clock} = 1)$
t_{in_R2}	$(P_{in\_Clock} = 1)$
t_{Clock}	$(P_{R1} = 1)$ OR $(P_{R2} = 1)$
$t_{Restart}$	$(P_{R1} = 0)$ AND $(P_{R2} = 0)$

Figure 6 presents the aging drone processing models without a restart, while Table 8 presents the legend for these models. This model is similar to the model presented in Figure 5. The difference is that we removed the restart mechanism and updated the battery model. The update of the battery model is required because we needed to include the probability of the aging states represented by the transitions t_{Aidle_w}, t_{Aidle_l}, t_{Abusy_l}, and t_{Abusy_l}. It is important to note that in the scenario with a restart, the battery model does not account for transitions representing the probability of aging states. This is because we assume the application is promptly restarted as soon as an aging state is detected.



(a) Drone model (b) Battery model

Fig. 6. Aging drone processing models without a restart.

5.4 Adopted metrics

In this study, we consider several key metrics: steady-state availability (AV), throughput (TP), drone flight autonomy (FA), and out-of-battery probability (OBP). Table 9 displays the expressions used to evaluate these metrics, which are defined for the models presented earlier in this work. It is worth noting that AV and TP can be computed using the lower-level component models, such as the drone model and network model. Note that $Dser$ and $Fser$ represent values assigned to transitions T_{D1_s} and T_{F_s}, respectively. Furthermore, we employ AV1 and AV2 to compute the steady-state availability of the AD model, both with and without restart mechanisms.

Autonomy-related metrics like FA and OBP need to be analyzed through a higher-level model, specifically, the battery model. The FA metric is calculated by determining the average time to absorption in $P_{S_discharge}$ within the battery model. In other words, it represents the average time it takes for the battery to run out. Unlike the other metrics in Table 9, FA is not explicitly defined as an expression because its value can be computed from the mean time to absorption within the model. On the other hand, the value of the OBP metric is calculated by the probability of battery discharge through a transient evaluation at a given time of the metric $P\{\#P_{S\_discharge} = 100\}$.

6 EVALUATION

To evaluate the proposed models and FA-Assure, we consider a case study of a SAR mission scenario and conduct numerical studies to derive the autonomy assurance strategy. First, we present a SAR scenario to illustrate the proposed

Table 8. Legend for the drone processing models without a restart.

Model	Transition	Description
Drone	P_{AD1_i}	Drone aging idle state
	P_{AD1_p}	Drone aging busy state
	T_{AD1_i}	Drone aging job arrival time
	T_{AD1_p}	Drone aging service time
	T_{A1}	Drone aging time in idle state
	T_{A2}	Drone aging time in busy state
	T_{D1_f3}	Drone failure time in aging busy state
	T_{D1_f4}	Drone failure time in aging idle state
	P_{D1_i}	Drone idle state
	P_{D1_p}	Drone busy state
	P_{D1_dw}	Drone down state
	T_{D1_j}	Drone job arrival time
	T_{D1_s}	Drone service time
	T_{D1_f1}	Drone failure time in busy state
	T_{D1_f2}	Drone failure time in idle state
	T_{D1_r}	Drone recovery time
Battery	P_{B_level}	Battery level state
	$P_{S\_discharge}$	Battery discharge state
	T_{dec}	Decision point
	P_{dec}	Decision state
	t_{Aidle_w}	Drone aging probability (Wi-Fi and idle)
	t_{Aidle_l}	Drone aging probability (LTE and idle)
	t_{Abusy_w}	Drone aging probability (Wi-Fi and busy)
	t_{Abusy_l}	Drone aging probability (LTE and idle)
	P_{Aidle_wC}	Battery aging discharge state (Wi-Fi and idle)
	P_{Aidle_lC}	Battery aging discharge state (LTE and idle)
	P_{Abusy_wC}	Battery aging discharge state (Wi-Fi and busy)
	P_{Abusy_lC}	Battery aging discharge state (LTE and busy)
	T_{Aidle_wC}	Battery aging discharge time (Wi-Fi and idle)
	T_{Aidle_lC}	Battery aging discharge time (LTE and idle)
	T_{Abusy_wC}	Battery aging discharge time (Wi-Fi and busy)
	T_{Abusy_lC}	Battery aging discharge time (LTE and busy)
	t_{idle_w}	Drone probability (Wi-Fi and idle)
	t_{idle_l}	Drone probability (LTE and idle)
	t_{busy_w}	Drone probability (Wi-Fi and busy)
	t_{busy_l}	Drone probability (LTE and idle)
	P_{idle_w}	Battery discharge state (Wi-Fi and idle)
	P_{idle_l}	Battery discharge state (LTE and idle)
	P_{busy_w}	Battery discharge state (Wi-Fi and busy)
	P_{busy_l}	Battery discharge state (LTE and busy)
	T_{idle_wC}	Battery discharge time (Wi-Fi and idle)
	T_{idle_lC}	Battery discharge time (LTE and idle)
	T_{busy_wC}	Battery discharge time (Wi-Fi and busy)
	T_{busy_lC}	Battery discharge time (LTE and busy)

Table 9. Expression for evaluating the adopted metrics.

Env.	Metric	Expression
DP	AV	P{((#P_{D1_i} = 1) OR (#P_{D1_p} = 1)) AND ((#P_{W_up} = 1) AND (#P_{W_act} = 1)) OR ((#P_{L_up} = 1) AND (#P_{L_act} = 1))}
	TP	$((E\{\#P_{D1\_p}\}) * (1/Dser))$
	OBP	$P\{\#P_{S\_discharge} = 100\}$
FP	AV	P{((#P_{D1_i} = 1) OR (#P_{D1_p} = 1)) AND ((#P_{F_b} = 1) OR (#P_{F_p} = 1)) AND ((#P_{W_up} = 1) AND (#P_{W_act} = 1)) OR ((#P_{L_up} = 1) AND (#P_{L_act} = 1))}
	TP	$((E\{\#P_{F\_p}\}) * (1/Fser))$
	OBP	$P\{\#P_{S\_discharge} = 100\}$
AD	AV1	P{((#P_{D1_i} = 1) OR (#P_{D1_p} = 1)OR (#P_{AD1_i} = 1) OR (#P_{AD1_p} = 1) OR (#P_{R1} = 1) OR (#P_{R2} = 1)) AND ((#P_{W_up} = 1) AND (#P_{W_act} = 1)) OR ((#P_{L_up} = 1) AND (#P_{L_act} = 1))}
	AV2	P{((#P_{D1_i} = 1) OR (#P_{D1_p} = 1)OR (#P_{AD1_i} = 1) OR (#P_{AD1_p} = 1)) AND ((#P_{W_up} = 1) AND (#P_{W_act} = 1)) OR ((#P_{L_up} = 1) AND (#P_{L_act} = 1))}
	OBP	$P\{\#P_{S\_discharge} = 100\}$

approach for modeling and analyzing a UAV in a mission-critical scenario. Following that, we present results related to availability, performance, and autonomy, particularly in the context of heavy-loaded processes (DP and FP). We then proceed to present the results pertaining to software aging processes, focusing on restarting as one of the recovery methods that can be implemented to mitigate application aging. Finally, we provide an analysis of the strategy.

For the sake of reproducibility, we divided the input parameter values in Tables 10, 11, and 12. In Table 10, we present the input parameter values for DP and FP models presented in Figures 3 and 4, respectively. In Tables 11 and 12, we present additional input parameter values for the aging drone processing models with and without a restart presented, respectively, in Figures 5 and 6. Note these input parameter values are assigned to the transitions of the SPN models for the purpose of numerical analysis. Despite the fact the parameter values can be measured or estimated from experiments, we gathered them from other works [17, 18]. The battery discharge values are based on [21]. For constructing our hierarchical models based on SPNs, we utilized both SPNP and Mercury to facilitate the modeling and subsequent numerical analysis. More specifically, we used SPNP for the availability-performance analysis, while Mercury is used for flight autonomy analysis. SPNP is a tool for modeling and analyzing stochastic systems using stochastic Petri nets [13]. It is widely used to model complex systems and analyze their performance by incorporating randomness into transitions and events. Mercury [20], on the other hand, is integrated software that enables the creation and evaluation of reliability block diagrams, stochastic Petri nets, continuous time Markov chains, and energy flow models. It offers a graphical interface for model creation, supports formal analysis, and simulations, making it valuable for designing and evaluating complex systems.

6.1 UAV in a SAR scenario

We consider a SAR scenario to illustrate the proposed approach to model and analyze a UAV in a mission-critical scenario (see Figure 7). The UAV captures images of the area of a mission to search for missing persons after a natural disaster. If a person is found, the UAV communicates the coordinates to the ground station. Wi-Fi or LTE networks are used for communication purposes. If the Wi-Fi network is not working, the LTE network is used instead. An object detection program runs on the drone to detect persons to rescue, but the process can be offloaded to a fog node in the

Table 10. Default input parameters for DP and FP models.

Transition	Variable	Value
T_{D1_j}	θ_{D1_j}	0.00139 (hrs)
T_{D1_s}	θ_{D1_s}	0.00083 (hrs)
T_{D1_f1}	θ_{D1_f1}	72 (hrs)
T_{D1_f2}	θ_{D1_f2}	336 (hrs)
T_{D1_r}	θ_{D1_r}	0.332 (hrs)
T_{F_n}	θ_{F_n}	0.000139 (hrs)
T_{F_s}	θ_{F_s}	0.00069 (hrs)
T_{F_f1}	θ_{F_f1}	730 (hrs)
T_{F_f2}	θ_{F_f2}	2920 (hrs)
T_{F_r}	θ_{F_r}	0.5 (hrs)
T_{W_f}	θ_{W_f}	2 (hrs)
T_{W_r}	θ_{W_r}	0.0028 (hrs)
T_{L_f}	θ_{L_f}	2 (hrs)
T_{L_r}	θ_{L_r}	0.0028 (hrs)
T_{W_act}	θ_{W_act}	0.000139 (hrs)
T_{L_act}	θ_{L_act}	0.000139 (hrs)
T_{idle_wC}	θ_{idle_wC}	0.09 (hrs)
T_{idle_lC}	θ_{idle_lC}	0.071666 (hrs)
T_{busy_wC}	θ_{busy_wC}	0.045 (hrs)
T_{busy_lC}	θ_{busy_lC}	0.035833 (hrs)
t_{idle_w}	θ_{idle_w}	0.142
t_{idle_l}	θ_{idle_l}	2.84E-05
t_{busy_w}	θ_{busy_w}	0.853
t_{busy_l}	θ_{busy_l}	0.00017

Table 11. Default input parameters for the aging processing models with a restart.

Transition	Variable	Value
T_{A1}	θ_{A1}	4 (hrs)
T_{A2}	θ_{A2}	4 (hrs)
T_{R1}	θ_{R1}	0.01666 (hrs)
T_{R2}	θ_{R2}	0.01666 (hrs)
T_{D1_f3}	θ_{D1_f3}	36 (hrs)
T_{D1_f4}	θ_{D1_f4}	168 (hrs)
t_{idle_w}	θ_{idle_w}	0.14
t_{idle_l}	θ_{idle_l}	0.84
t_{busy_w}	θ_{busy_w}	1.86E-4
t_{busy_l}	θ_{busy_l}	0.001

fog node infrastructure. FA-Assure is used for determining the right condition to start or stop computation offloading depending on internal and external system states or when restarting faulty application processes.

6.2 Performance and availability analysis

The stability of a Wi-Fi connection for a flying drone can vary significantly depending on the location. For example, UAVs in a SAR mission may need to connect to multiple Wi-Fi hotspots as they cover a given area. Therefore, in Figure 8,

Table 12. Default input parameters for the aging processing models without a restart.

Transition	Variable	Value
T_{Aidle_wC}	θ_{Aidle_wC}	0.045 (hrs)
T_{Aidle_lC}	θ_{Aidle_lC}	0.035 (hrs)
T_{Abusy_wC}	θ_{Abusy_wC}	0.0225 (hrs)
T_{Abusy_lC}	θ_{Abusy_lC}	0.017 (hrs)
t_{Aidle_w}	θ_{Aidle_w}	0.58
t_{Aidle_l}	θ_{Aidle_l}	7.78E-4
t_{Abusy_w}	θ_{Abusy_w}	0.35
t_{Abusy_l}	θ_{Abusy_l}	4.67E-4
t_{idle_w}	θ_{idle_w}	0.0073
t_{idle_l}	θ_{idle_l}	9.5E-6
t_{busy_w}	θ_{busy_w}	0.043
t_{busy_l}	θ_{busy_l}	5.71E-5

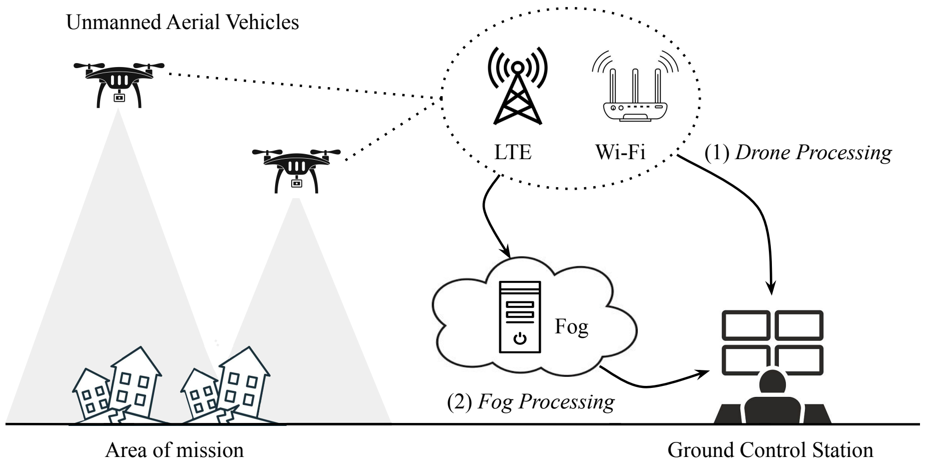


Fig. 7. UAVs in a mission-critical scenario.

we plot the impact of the mean time between Wi-Fi disconnections on the availability. On the x-axis we vary the mean time between Wi-Fi disconnections (θ_{W_f}), starting from 5 up to 300 minutes. If we consider the mean time between Wi-Fi disconnections less than 60 minutes, the results show that having both Wi-Fi and LTE network connections has better results than just one in terms of availability. For the cases where the mean time between Wi-Fi disconnections is bigger than 60 minutes, the availability for DP and FP starts to get very close. However, FP still has the best levels of availability. These results revealed that for critical missions where the mean time between Wi-Fi disconnections is low, the best option is always to use more than one network connection. Nevertheless, as the mean time between Wi-Fi disconnections increases, FP is the best option and the choice of whether or not to use redundant networks is not so significant.

Figure 9 presents the impact of the mean time between Wi-Fi disconnections on the throughput. Note that high throughput is very important for mission-critical UAVs since they need to be able to process large amounts of data

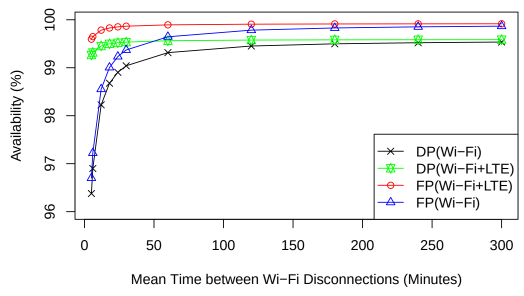


Fig. 8. Availability analysis.

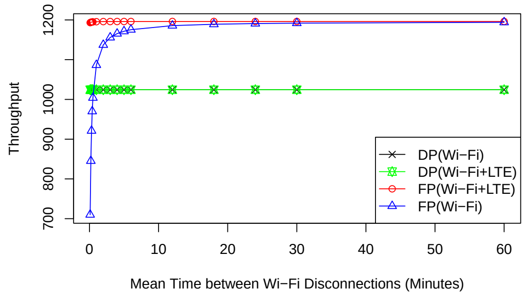


Fig. 9. Throughput analysis.

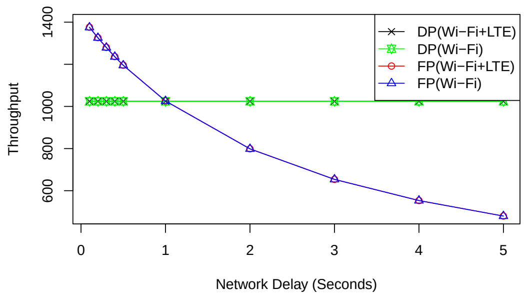


Fig. 10. Network delay analysis.

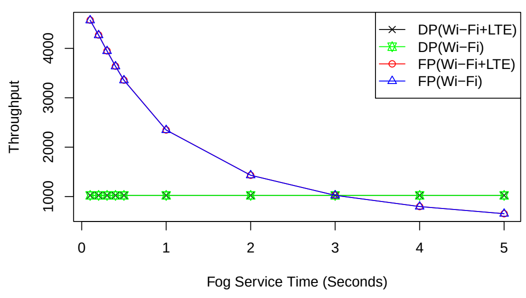


Fig. 11. Fog processing time analysis.

quickly. For instance, high throughput rates enable the timely delivery of videos or images of a rescue area, which is crucial for the success of any SAR mission. We vary the θ_{W_f} from 0.1 up to 60 minutes on the x-axis. As the image processing for DP is not affected by the network because the image processing is carried out locally, the results considering only Wi-Fi and both (Wi-Fi and LTE network connections) are the same (see green and black lines). The throughput of FP, on the other hand, is highly affected, especially, when the mean time between Wi-Fi disconnections is less than 6 minutes and only a Wi-Fi configuration is considered. Nevertheless, the longer the mean time between Wi-Fi disconnections, the higher the throughput for FP. In short, if the network connection is unstable, the best option is FP(Wi-Fi+LTE). On the other hand, if the network connection is stable, the best option is FP using only Wi-Fi or both Wi-Fi and LTE.

It is known that one of the main advantages of using fog computing is to decrease the latency by processing selected data locally, instead of sending it to the cloud for analysis. However, it will depend on how far away the fog is located and how much data needs to be sent to it. Additionally, fog nodes may not always be available for a UAV in mission-critical scenarios. Therefore, Figure 10 presents the impact of network delays on throughput. We vary the fog network delay (θ_{F_n}) from 0.1 to 5 seconds. The results confirm that FP is highly affected by network delays. As the network delay increases, the throughput decreases for FP. On the other hand, DP is not affected by long-distance network delays because there is no fog node for this environment. Also, the throughput difference for FP using only Wi-Fi and both Wi-Fi and LTE is very small.

As stated by [3], "fog nodes can be any device with computing, storage, and network connectivity and are deployed anywhere with a network connection". Nevertheless, these devices have different processing times and can affect SAR missions. Therefore, Figure 11 presents the impact of the mean fog service time on the throughput. We vary the mean fog processing time (θ_{F_s}) from 0.1 to 5 seconds. Note that the throughput for DP is constant because there is no fog node for this environment and the throughput difference for FP using only Wi-Fi and both Wi-Fi and LTE is very small. As expected for FP, the shorter the Fog processing time is, the better the throughput is. It also reveals that the throughput of DP is only better than FP when the mean fog service time is greater than 3 seconds.

6.3 Flight autonomy analysis

One of the concerns in a UAV operation is the battery life which limits the length of the flight. In Table 13, we present the FA in terms of battery lifetime, taking into account distinct network connections. The results show that FP is better than DP for all the scenarios. LTE is turned out to be the worst option in terms of the battery lifetime regardless of computation modes because the LTE connection has a higher discharge rate than Wi-Fi.

Table 13. Battery Lifetime for Different Scenarios.

Scenario	WIFI	LTE	WIFI+LTE
DP	5.181	4.126	5.180
FP	9.000	7.167	8.998

UAV pilots who are willing to plan and execute a mission should be aware of the UAV flight autonomy. The CDF (Cumulative Distributed Function) of OBP indicates the probability of a UAV running out of battery in the air and crashing to the ground. Note that we compute the CDF by a transient evaluation of the metric $P\{\#P_{S\_discharge} = 100\}$. In Figure 12, we present the CDFs for DP considering Wi-Fi, Wi-Fi plus LTE, and LTE. Note that the difference between Wi-Fi plus LTE and just Wi-Fi is minimal so the lines are overlapping each other. The results indicate that the average probabilities of a UAV running out of battery within 5 hours are 38.98%, 38.89%, and 96.76% for Wi-Fi, Wi-Fi plus LTE, and LTE, respectively. Nevertheless, the probability increases with time, so that at 8.3 hours the probability of a UAV running out of battery considering Wi-Fi and Wi-Fi plus LTE is 100%, while at 6.6 hours the same probability considering LTE is 100%.

In Figure 13, we compare the CDFs considering distinct network connections for FP. Similar to DP, the difference between Wi-Fi plus LTE and just Wi-Fi is minimal, therefore the lines are overlapping each other. Compared to DP, FP has much better flight autonomy. For instance, the results indicate that the average probability of a UAV running out of battery within 8 hours is 13.15%, 13.09%, and 87.57% for Wi-Fi, Wi-Fi plus LTE, and LTE, respectively. On the other hand, the UAV is out of battery at 13.7 hours for Wi-Fi and Wi-Fi plus LTE and at 11.1 for the LTE.

Drone battery consumption can be highly affected by external factors like wind speed and direction [39]. Therefore, we compare the CDFs of DP and FP modes for Wi-Fi plus LTE scenario considering three conditions (normal, medium, and harsh). For the medium and harsh conditions, we simulated scenarios where the battery consumption rates were accelerated by 50% and 100%, respectively, compared to the normal condition (θ_{idle_wC}, θ_{idle_lC}, θ_{busy_wC} and θ_{busy_lC}). In the medium condition, where drones may encounter moderately adverse conditions such as increased wind resistance or challenging flight conditions, the battery depletes 1.5 times faster than under normal circumstances. On the other hand, the harsh condition represents the most challenging operational scenario, characterized by extreme environmental conditions that impose substantial energy demands on drones, such as strong winds or turbulent air. In this scenario, we assumed a 100% increase in battery consumption rate compared to the normal condition, causing the battery to deplete at twice the rate it would under normal circumstances. Figures 14 and 15 present CDFs considering the three conditions.

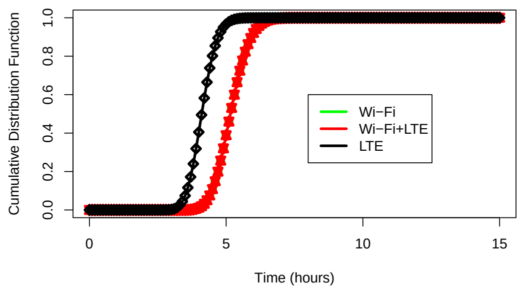


Fig. 12. CDFs of OBP for DP under different networks.

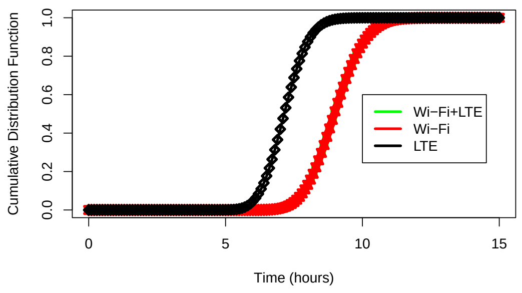


Fig. 13. CDFs of OBP for FP under different networks.

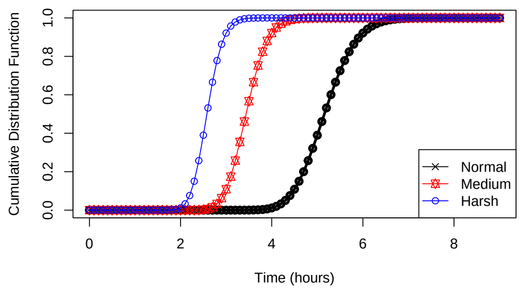


Fig. 14. CDFs of OBP for DP under different conditions.

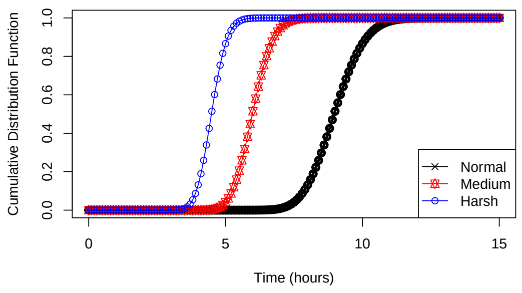


Fig. 15. CDFs of OBP for FP under different conditions.

For DP, the results show the UAV is out of battery at 8.3, 5.6, and 4.2 hours for normal, medium, and harsh conditions. For FP, the results show the UAV is out of battery at 13.7, 9.3, and 7.0 hours for normal, medium, and harsh conditions. The only scenario where DP is better than FP is when the environmental condition for DP is normal and that for FP is medium or harsh. The results show that environmental conditions can have a great impact on the probability of a UAV to fulfill its mission.

6.4 Aging drones analysis

As presented before, application programs are subject to software aging like classification systems. These classification systems are fundamental for SAR missions because they can be used for identifying potential victims in a rescue area. Restarting is one of the recovery methods that can be implemented to prevent applications from aging. However, it is important to note that restarting can be costly and lead to application downtime. Thus, assessing this recovery method is fundamental.

In Table 14, we present a comparative analysis of availability, considering scenarios both with and without restarts. In the with restart scenario, the availability slightly surpasses that of the w/o restart scenario, with values of 0.9959 and 0.9953, respectively. This implies that the strategy involving periodic restarts offers advantages in terms of availability, as the application is promptly restarted upon the detection of aging. The downtime column quantifies the total downtime experienced by each scenario over the course of a year. The w/o restart scenario accumulates more annual downtime, totaling 40.776 hours, compared to the with restart scenario, which experiences 35.563 hours of downtime. This suggests that the with restart scenario experiences less downtime throughout the year, attributed to its prompt restarts upon aging detection.

Table 14. Availability and downtime analysis.

Scenario	Availability	Annual Downtime (hours)
With restart	0.9959	35.563
W/o restart	0.9953	40.776

In Figure 16, we compare the CDFs of the OBP for a UAV considering with and without restart. Compared to the UAV without restart, the UAV with restart has much better flight autonomy. For instance, the results indicate that the average probability of a UAV running out of battery within 4 hours is 98.95%, and 1.14% for the cases without and with restart, respectively. Additionally, the UAV is out of battery at 8.3 hours for the case with restart and at 5 hours for the case without restart. These results show that restarting critical aging applications can increase UAV autonomy.

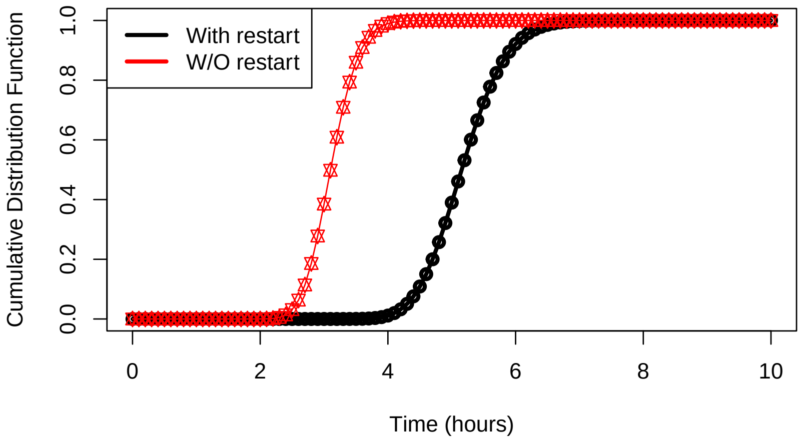


Fig. 16. CDFs of OBP for a UAV with and without restart.

Our analysis of aging drones has highlighted the importance of addressing software aging in critical applications like classification systems used in search and rescue missions. While the use of periodic restarts comes with some cost and potential downtime, our findings show that this strategy can significantly enhance availability and considerably extend UAV flight autonomy. This emphasizes the significance of employing SPN models to analyze UAV systems, ensuring the maintenance of availability and performance for these crucial UAV systems that play an essential role in mission success and saving lives.

6.5 Strategy analysis

The results of sensitivity analysis can guide the design of the UAV system that contains the strategy to mitigate faulty application programs. One of the important aspects of the strategy is whether the system can effectively use computation offloading under given environments. From the performance and availability analysis, we observe that the benefit of FP mode is reduced by a longer fog network delay and a larger fog service time. On the other hand, FP mode is always preferable if a higher FA is required, as presented in the flight autonomy analysis. If the environmental conditions discourage offloading, the system needs to operate in DP mode. In this case, the impact of software aging in the application process must be considered. As shown in the recovery operation analysis, application program restart should be employed if FA needs to be larger than approximately 2.5 hours. Based on these observations, the system engineers may derive a strategy like presented in Figure 17.

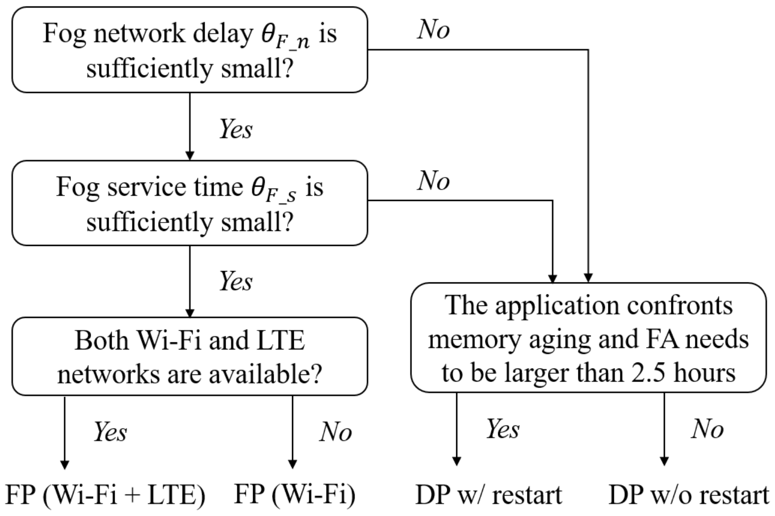

Fog network delay θ_{F_n} is sufficiently small?
No
Yes
Fog service time θ_{F_s} is sufficiently small?
No
Yes
Both Wi-Fi and LTE networks are available?
The application confronts memory aging and FA needs to be larger than 2.5 hours
Yes
No
Yes
No
FP (Wi-Fi + LTE)
FP (Wi-Fi)
DP w/ restart
DP w/o restart

Fig. 17. A strategy to select the computation mode for autonomy assurance.

If fog network delay θ_{F_n} and fog service time θ_{F_s} are sufficiently small ($\theta_{F_n} < 1s$ and $\theta_{F_s} < 3s$ in our numerical setting), choosing FP mode can be the best option for systems requiring high FA. If fog network delay or fog service time is larger than a certain threshold value (i.e., $\theta_{F_n} > 1$ or $\theta_{F_s} > 3$), DP mode can be a better choice (See Figure. 10 and 11 as well). However, in this case, the impact of a faulty application process running on the UAV should be considered as well. If the application process confronts software aging and FA is required to be larger than 4 hours, a restarting method needs to be implemented. The derived strategy has a certain level of generality and may be useful in a higher-level system design. However, the system parameter values differ in other application scenarios. FA-Assure can assist engineers to conduct such sensitivity analysis and to derive the appropriate strategy for given missions.

7 CONCLUSION

In this work, we presented FA-Assure as an autonomy assurance framework for UAV systems in mission-critical scenarios based on a performability analysis approach. FA-Assure employs hierarchically-configured SPNs to model and assess flight autonomy performance subject to faulty application processes. Our approach allows rescue teams to choose the relevant strategy to ensure the required flight autonomy to accomplish the mission. Through the numerical study on a SAR mission case study, we showed an example of a strategy for determining computation offloading for given environmental conditions. However, it is important to note that the challenges in the field of drone technology extend beyond single-drone operations. As the present work only considers a single drone, in our future work, we plan to extend our research to address these challenges by considering a swarm of drones where multiple drones collaboratively work together for mission-critical tasks. Our intention is to explore key issues such as swarm behavior, scalability, safety, energy efficiency, and practical applications in the context of drone swarms. These challenges represent critical factors in the development and deployment of drone swarm technology in mission-critical scenarios. By addressing these challenges, we aim to contribute to the advancement and practical implementation of drone swarm technology, enabling its effective use in a wide range of mission-critical applications. Our work also focused on the early stage of system design and assumed exponential distributions for state transitions in the proposed SPNs. Collecting data from real UAV environments to estimate realistic distributions is a necessary step toward the detailed design of the system. Thus, we plan to collect this data and investigate how the changes in distributions affect the autonomy evaluation.

ACKNOWLEDGMENTS

The authors would like to thank Kengo Watanabe and Qinyang Zhang to help produce the experiment results discussed in Section 3.

REFERENCES

[1] Marco Ajmone Marsan, Gianni Conte, and Gianfranco Balbo. 1984. A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems. *ACM Transactions on Computer Systems (TOCS)* 2, 2 (1984), 93–122.

[2] Ermeson Andrade, Fumio Machida, Roberto Pietrantuono, and Domenico Cotroneo. 2020. Software Aging in Image Classification Systems on Cloud and Edge. In *2020 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*. IEEE, 342–348.

[3] João Bachiega Jr, Breno Costa, and Aleteia PF Araujo. 2022. Computational Perspective of the Fog Node. *arXiv preprint arXiv:2203.07425* (2022).

[4] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. 2012. Fog computing and its role in the internet of things. *Proceedings of the first edition of the MCC workshop on Mobile cloud computing* (2012), 13–16.

[5] Hwei-Ming Chung, Sabita Maharjan, Yan Zhang, Frank Eliassen, and Kai Strunz. 2020. Placement and routing optimization for automated inspection with unmanned aerial vehicles: a study in offshore wind farm. *IEEE Transactions on Industrial Informatics* 17, 5 (2020), 3032–3043.

[6] Gianfranco Ciardo, Jogesh K Muppala, Kishor S Trivedi, et al. 1989. SPNP: Stochastic Petri Net Package.. In *PNPM*, Vol. 89. Citeseer, 142–151.

[7] Domenico Cotroneo, Francesco Fucci, Antonio Ken Iannillo, Roberto Natella, and Roberto Pietrantuono. 2016. Software aging analysis of the android mobile os. In *2016 IEEE 27th international symposium on software reliability engineering (ISSRE)*. IEEE, 478–489.

[8] Graham R Drozeski. 2005. *A fault-tolerant control architecture for unmanned aerial vehicles*. Georgia Institute of Technology.

[9] Emad Ebeid, Martin Skriver, Kristian Husum Terkildsen, Kjeld Jensen, and Ulrik Pagh Schultz. 2018. A survey of Open-Source UAV flight controllers and flight simulators. *Microprocessors and Microsystems* 61 (2018), 11–20. https://doi.org/10.1016/j.micpro.2018.05.002

[10] Michael A Goodrich, Bryan S Morse, Damon Gerhardt, Joseph L Cooper, Morgan Quigley, Julie A Adams, and Curtis Humphrey. 2008. Supporting wilderness search and rescue using a camera-equipped mini UAV. *Journal of Field Robotics* 25, 1-2 (2008), 89–110.

[11] Michael Grottke, Rivalino Matias, and Kishor S Trivedi. 2008. The fundamentals of software aging. In *2008 IEEE International conference on software reliability engineering workshops (ISSRE Wksp)*. Ieee, 1–6.

[12] Lav Gupta, Raj Jain, and Gabor Vaszkun. 2015. Survey of important issues in UAV communication networks. *IEEE Communications Surveys & Tutorials* 18, 2 (2015), 1123–1152.

[13] Christophe Hirel, Bruno Tuffin, and Kishor S Trivedi. 2000. Spnp: Stochastic petri nets. version 6.0. In *Computer Performance Evaluation. Modelling Techniques and Tools: 11th International Conference, TOOLS 2000 Schaumburg, IL, USA, March 27–31, 2000 Proceedings 11*. Springer, 354–357.

[14] Shahrear Iqbal. 2021. A Study on UAV Operating System Security and Future Research Challenges. In *2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC)*. 0759–0765.

[15] Hao Jiang, Hongli Yang, Shengchao Qin, Zhendong Su, Jian Zhang, and Jun Yan. 2017. Detecting energy bugs in android apps using static analysis. In *International Conference on Formal Engineering Methods*. Springer, 192–208.

[16] Anamta Khan, Naghmeh Ivaki, and Henrique Madeira. 2022. Are UAVs' Flight Controller Software Reliable?. In *2022 IEEE 27th Pacific Rim International Symposium on Dependable Computing (PRDC)*. IEEE, 194–204.

[17] Fumio Machida and Ermeson Andrade. 2021. Availability Modeling for Drone Image Processing Systems with Adaptive Offloading. In *2021 IEEE 26th Pacific Rim International Symposium on Dependable Computing (PRDC)*. IEEE, 93–103.

[18] Fumio Machida and Ermeson Andrade. 2021. PA-Offload: performability-aware adaptive fog offloading for drone image processing. In *2021 IEEE 5th International Conference on Fog and Edge Computing (ICFEC)*. IEEE, 66–73.

[19] Fumio Machida, Qingyang Zhang, and Ermeson Andrade. 2023. Performability analysis of adaptive drone computation offloading with fog computing. *Future Generation Computer Systems* 145 (2023), 121–135. https://doi.org/10.1016/j.future.2023.03.027

[20] Paulo Maciel, Rubens Matos, Bruno Silva, Jair Figueiredo, Danilo Oliveira, Iure Fé, Ronierison Maciel, and Jamilson Dantas. 2017. Mercury: Performance and dependability evaluation of systems with exponential, expolynomial, and general distributions. In *2017 IEEE 22nd Pacific Rim international symposium on dependable computing (PRDC)*. IEEE, 50–57.

[21] Rubens Matos, Jean Araujo, Danilo Oliveira, Paulo Maciel, and Kishor Trivedi. 2015. Sensitivity analysis of a hierarchical model of mobile cloud computing. *Simulation Modelling Practice and Theory* 50 (2015), 151–164.

[22] Arvind Merwaday and Ismail Guvenc. 2015. UAV assisted heterogeneous networks for public safety communications. In *2015 IEEE wireless communications and networking conference workshops (WCNCW)*. IEEE, 329–334.

[23] Mohamed-Ayoub Messous, Sidi-Mohammed Senouci, Hichem Sedjelmaci, and Soumaya Cherkaoui. 2019. A game theory based efficient computation offloading in an UAV network. *IEEE Transactions on Vehicular Technology* 68, 5 (2019), 4964–4974.

[24] Michael K. Molloy. 1982. Performance analysis using stochastic Petri nets. *IEEE Transactions on computers* 31, 09 (1982), 913–917.

[25] Tuan Anh Nguyen, Kwonsu Jeon, Jae-Woo Lee, Iure Fe, and Francisco Airton Silva. 2022. Model-driven Mission Dependability Design of Unmanned Aerial Systems. In *AIAA AVIATION 2022 Forum*. 4091.

[26] Pijush Kanti Dutta Pramanik, Nilanjan Sinhababu, Bulbul Mukherjee, Sanjeevikumar Padmanaban, Aranyak Maity, Bijoy Kumar Upadhyaya, Jens Bo Holm-Nielsen, and Prasenjit Choudhury. 2019. Power consumption analysis, measurement, management, and issues: A state-of-the-art review of smartphone battery and energy usage. *IEEE Access* 7 (2019), 182113–182172.

[27] Supravat Samanta, Subhajit Pramanick, and Partha Sarathi Mandal. 2021. Fault-Tolerant Covering Points by UAVs. In *8th International Conference on Networking, Systems and Security*. 60–64.

[28] David C Schedl, Indrajit Kurmi, and Oliver Bimber. 2021. An autonomous drone for search and rescue in forests using airborne optical sectioning. *Science Robotics* 6, 55 (2021), eabg1188.

[29] Tabea Schmidt and Alexander Pretschner. 2022. StellaUAV: A Tool for Testing the Safe Behavior of UAVs with Scenario-Based Testing. In *IEEE 33rd International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 37–47.

[30] Mahmood Shafiee, Zeyu Zhou, Luyao Mei, Fateme Dinmohammadi, Jackson Karama, and David Flynn. 2021. Unmanned aerial drones for inspection of offshore wind turbines: A mission-critical failure analysis. *Robotics* 10, 1 (2021), 26.

[31] Hazim Shakhatreh, Ahmad H Sawalmeh, Ala Al-Fuqaha, Zuochao Dou, Eyad Almaita, Issa Khalil, Noor Shamsiah Othman, Abdallah Khreishah, and Mohsen Guizani. 2019. Unmanned aerial vehicles (UAVs): A survey on civil applications and key research challenges. *Ieee Access* 7 (2019), 48572–48634.

[32] Mario Silvagni, Andrea Tonoli, Enrico Zenerino, and Marcello Chiaberge. 2017. Multipurpose UAV for search and rescue operations in mountain avalanche events. *Geomatics, Natural Hazards and Risk* 8, 1 (2017), 18–33.

[33] Tullio Joseph Tanzi, Madhu Chandra, Jean Isnard, Daniel Camara, Olivier Sébastien, and Fanilo Harivelo. 2016. Towards" drone-borne" disaster management: future application scenarios. In *XXIII ISPRS Congress, Commission VIII (Volume III-8)*, Vol. 3. Copernicus GmbH, 181–189.

[34] Max Taylor, Jayson Boubin, Haicheng Chen, Christopher Stewart, and Feng Qin. 2021. A Study on Software Bugs in Unmanned Aircraft Systems. In *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*. 1439–1448. https://doi.org/10.1109/ICUAS51884.2021.9476844

[35] Panagiotis Vekris, Ranjit Jhala, Sorin Lerner, and Yuvraj Agarwal. 2012. Towards verifying android apps for the absence of no-sleep energy bugs. In *Presented as part of the 2012 Workshop on Power-Aware Computing and Systems*.

[36] Dinghua Wang, Shuqing Li, Guanping Xiao, Yepang Liu, and Yulei Sui. 2021. An exploratory study of autopilot software bugs in unmanned aerial vehicles. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 20–31.

[37] Kengo Watanabe, Fumio Machida, Ermeson Andrade, Roberto Pietrantuono, and Domenico Cotroneo. 2023. Software Aging in Real-Time Object Detection Systems on Edge Servers. In *Proceedings of the 38th Annual ACM Symposium on Applied Computing*. 671–678.

[38] Mingyuan Xia, Wenbo He, Xue Liu, and Jie Liu. 2013. Why application errors drain battery easily? A study of memory leaks in smartphone apps. In *Proceedings of the Workshop on Power-Aware Computing and Systems*. 1–5.

[39] Connie Zeng, Jacob Knickerbocker, Razia Shaik, and Kevin Marx. 2020. Routing of hitchhiking drones with respect to autonomous and connected vehicles. US Patent App. 16/756,582.

[40] Juan Zhang, James F Campbell, Donald C Sweeney II, and Andrea C Hupman. 2021. Energy consumption models for delivery drones: A comparison and assessment. *Transportation Research Part D: Transport and Environment* 90 (2021), 102668.