DEPENDABILITY ANALYSIS METHODOLOGY FOR FPGA-BASED UAV COMMUNICATION PROTOCOLS USING UPPAAL-SMC

Mohamed Abdelhamid Mohamed Abdelhamid

A thesis
in
The Department
of
Computer and Electrical Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of
Master of Applied Science (electrical and computer engineering) at
Concordia University
Montréal, Québec, Canada

January 2022

Concordia University
School of Graduate Studies

This is to certify that the thesis prepared

By: **Mohamed Abdelhamid Mohamed Abdelhamid**

Entitled: **Dependability Analysis Methodology for FPGA-based UAV Communication Protocols using UPPAAL-SMC**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Electrical and Computer Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining commitee:

_____ Chair
Dr. Abdelwahab Hamou-Lhadj

_____ Examiner
Dr. Jamal Bentahar

_____ Examiner
Dr. Abdelwahab Hamou-Lhadj

_____ Thesis Supervisor(s)
Dr. Otmane Ait Mohamed

Approved by _____
Dr. Yousef R. Shayan, Chair of the ECE Department

_____
Dr. Mourad Debbabi, Dean,Faculty of Engineering and Computer Science

Date _____

Abstract

Dependability Analysis Methodology for FPGA-based UAV Communication Protocols using UPPAAL-SMC

Mohamed Abdelhamid Mohamed Abdelhamid
Concordia University 2022

UAVs are multifaceted devices that have enormous versatility and flexibility in a plethora of various fields. Year over year, UAVs see a tremendous amount of research invested in it to make them more efficient and autonomous when performing a task. This increase in autonomy requires the UAVs to have a dependable link between them to exchange crucial information like current position and speed. These messages are transmitted to avoid collisions and perform missions efficiently. The communication between UAVs depends on several factors like the used telemetry device, distance between the UAVs, speed of the UAVs, and application environment. Hence, an UAV designer must analyze the reliability of the communication based on the desired application environment and necessary communication components in the UAV.

Faults can also propagate in UAV components built using the FPGA technology when they are placed in harsh radiation environments like radiation monitoring. These errors can lead to complications in the operation of an UAV communication component, and hence, FPGAs require techniques like blind scrubbing to mitigate these faults. The availability of the communication component can be impacted when using this mitigation approach. Therefore, investigating the optimal configuration to maintain high and consistent availability is crucial.

This thesis presents a methodology to perform high-level dependability analysis for UAV communication protocols using statistical model checking. First, we evaluate the reliability of a point-to-point UAV communication using the UAV-UAV framework. The main objective of this framework is to investigate the link reliability between UAVs based on the specifications of the telemetry device and the availability of the communication components. To accomplish this, we propose models to emulate the behavior of two UAVs in air, the condition of the transmitter and receiver, and the

data exchange phase between two UAVs. Then, we analyze the availability of an UAV communication module in a harsh radiation environment using blind scrubbing as a mitigation approach. The peak availability of UAV-UAV and UAV-GCS communication components is investigated through the UAV-UAV and UAV-GCS frameworks. The two frameworks utilize the SEU rate computed from the RTL code of the communication component design. Then, implement crucial features like scrubbing interval and scrub time in the transmitter and receiver modules to find the optimal scrubbing interval when the UAV communications with other UAVs or the GCS. Finally, the effect of these faults and limitations of blind scrubbing is also investigated in our work.

To my Parents, Grandmother, Brother, and Sister whom supported me unconditionally.

Acknowledgments

Working on the thesis for my Master's degree in the Hardware Verification Group (HVG) at Concordia was a staggering experience that I learned a lot from personally and professionally. This is thanks to all the guidance and valuable feedback that I received from my supervisor Dr. Otmane Ait Mohamed. I want to thank Dr. Otmane Ait Mohamed for his constant engorgement and support. Throughout my thesis, he was very inspiring and present in every single step of my work. Researching with Dr.Ait Mohamed was a huge learning opportunity that will help further advance me in the future. Altogether I was able to proceed with my work because of his constant effort and aid.

I am also deeply indebted to Dr. Marwan Ammar and Dr. Ayman Attallah, who helped me in various stages of my research. Their insight and comments when approaching a problem in the research were extremely helpful throughout this thesis. I learned a lot from their consistent hard work and vast experience in this field.

I want to thank the team at HVG who welcomed me in the group and were very supportive, especially Alain Aoun, Yasmeen El-Derhalli, Hassnaa El-Derhalli, Abdel-Latif Alshalalfah, and Vivek Bansal. They all treated me with kindness and provided me with an overwhelming amount of aid whenever I needed it. Also, I would like to thank Rawan Genina for her unconditional believe and support. Finally, my friends Mazen Abdel-Razek and Mohammed AbuFaris were a massive source of encouragement and joy throughout my degree.

Last but definitely not least, I would like to say that I would not be where I am right now without my family's unconditional love and support. My father's comprehensive guidance and my mother's endless love and support were always there for me in every stage and step of my research. My family always showed me an immense amount of appreciation, and they were my pillars of support not only in this thesis but throughout my life. My brother and sister were always a source of humor and

pride for me, and they especially made my days during lockdown a lot easier.

Table of Content

List of Figures

List of Tables

List of Acronyms

UAV	Unmanned Aerial vehicle
GCS	Ground Command Station
SMC	Statistical Model Checking
PTA	Priced Timed Automata
FSM	Finite-State Machine
SEE	Single Event Effects
SEU	Single Event Upset
TMR	Triple Modular Redundancy
MAVlink	Micro Aerial Vehicle link
P_r	Received signal strength
P_t	Transmitted Power
G_r	Transmitter Gain
G_r	Receiver Gain
RTL	Register Transfer Level
SMT	Satisfiability Modulo Theory
FPGA	Field Programmable Gate Array
UART	Universal Asynchronous Receiver Transmitter
Gs	Ground Speed
U1	Replaced UAV
U2	Replacing UAV
P	SEU rate
λ_{device}	Device Failure rate

Chapter 1

Introduction

The following chapter provides a brief introduction to the UAV domain and highlights the contribution of our thesis in this field. After providing the context and motivation for our research, we discuss the various essential components of an UAV and the overall architecture of a design using SRAM-based FPGAs. Additionally, we present the dependability metrics used in our work. We also discuss the techniques utilized to evaluate these attributes and highlight several examples from the literature that implements them. Then, we provide a brief overview of radiation effects, mitigation approaches, and analysis techniques related to them. Subsequently, we present the problem statement followed by our thesis contributions. Finally, the introduction chapter is concluded with the outline for this thesis.

1.1 Context and Motivation

The Royal Flying Corps of the British army first introduced Unmanned Aerial Vehicle (UAV) during World War I [35]. Since then, the use of UAVs has evolved and expanded till it hit a market cap of almost 4 billion dollars in 2021. This market is expected to grow three folds to reach more than 12 billion dollars by the year 2025 [12]. Nowadays, the appeal of UAV vehicles mainly come from non-military applications. This predominantly is due to their versatility and efficiency in performing their assigned tasks. Recently, the use of UAV is researched in a plethora of projects because they can perform missions that are logistically impossible or critically time-consuming like humanitarian relief operations [17]. Also, companies study UAVs in

commercial fields because of their efficiency in performing the task [51]. Moreover, UAVs are placed in harsh environments to perform missions that can potentially be hazardous to humans like radiation monitoring.

All the aforementioned applications requires a dependable UAV link with the outside world. Therefore, communication is a fundamental aspect for any application that involve UAVs. It is vital because it acts as the gateway for it to talk with the outside world to relay and send sensitive information to other UAVs and the Ground Command Station (GCS). Sensitive information can include data collected from sensors, current position of objects in the environment to avoid collisions with the UAV, or messages to instruct them about their next task in a mission.

Having UAVs that communicate together is considerably more beneficial for several reasons. To begin with, using several UAVs is very effective and accurate for a mission because multiple UAVs perform micro-tasks simultaneously, making them time efficient [7]. Also, UAVs can cover significantly more areas due to the increase in the number of UAVs used in a mission [49]. This make them more autonomous compared to using a single UAV for a mission.

To achieve this autonomy, an UAV will have to communicate with the surrounding UAVs, GCS and other objects in the environment. Therefore, they will require more complex communication topologies due to the limited input provided by the user[49]. Several research papers were introduced to address the complexities regarding the communication between UAVs. These protocols are relatively new and hence, have limited dependability analysis performed on them.

Efficient and reliable UAV-UAV and UAV-GCS communication is crucial to maintain the functionality of an UAV. However, several limitations and factors like the decrease in the UAVs transmission power due to low battery life and hardware failure due to soft errors in a radiation environment can affect the communication link of an UAV. A failure in communication can lead to catastrophic events such as collision between UAVs or other objects in its vicinity. Additionally, such failure can cause delays in receiving and sending instructions to the UAV and hence, reduce the efficiency of the overall mission. Therefore, it is necessary to evaluate the effect of communication components and the application environment on the dependability of an UAV communication protocol.

1.2 UAV Architecture

In this section, we briefly overview various fundamental components currently utilized in any UAV. Then, we discuss how these components interact with each other. Finally, we present the design architecture of UAV components based on the FPGA technology.

1.2.1 Overall Design of an UAV

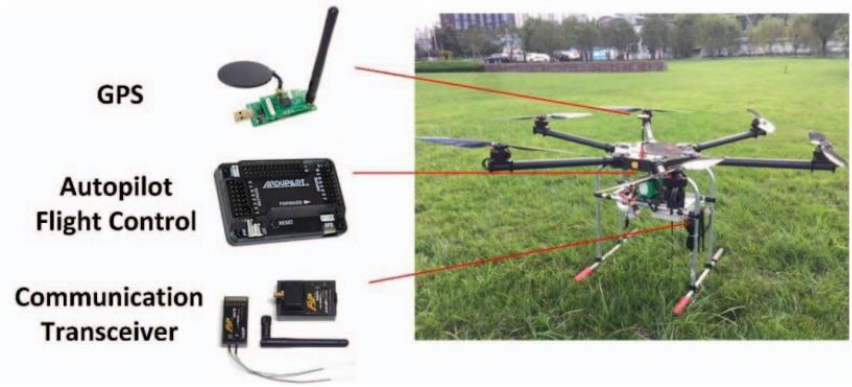

Figure 1.1: Overall UAV architecture [52].

Generally, an UAV is composed of the Global Positioning System (GPS), Autopilot flight control, and communication modules as shown in Figure1.1 [52]. First, the UAV uses the GPS to approximate its current position in the x, y, and z planes. This component is crucial for any UAV because it is needed to perform a task remotely and avoid collision with other UAVs in its proximity.

The autopilot flight control is the heart of an UAV, where all the necessary computations and decision-making occur. Initially, it collects details from the environment through many sensors like the inertial measurement sensor, which measures the angular velocity, and the ultra wave sensor to approximately calculate the distance between the UAV and another object. The flight controller takes all of these inputs into consideration to reach the optimal decision. This decision is then relayed to the microcontrollers of the UAV to move the motors accordingly.

Exchanging information with the outside world occurs through the transceiver

component using a communication protocol. UAV communication protocol characteristics differ from the other wireless protocols like the Mobile ad-hoc Networks (MANETS) and vehicular ad-hoc networks (VANETS) in its mobility, frequency of changes in topology, and more significant energy constraints for an UAV in a network [20].

The Micro Aerial Vehicle link (MAVlink) and the Mission Based UAV Swarm coordination (MUSCOP) protocols are two examples of a communication protocols built for UAV. The MAVlink protocol is an open-source payload-only communication protocol and is mainly used for single point-point communication between a GCS and an UAV [28]. The authors in [20] expanded the MAVlink protocol by adding extra messages to allow UAVs to collaborate by creating formations, swarms and perform swarm maintenance when needed [14]. The MUSCOP protocol was proposed in early 2020 by [18] to allow UAVs to coordinate using a master-slave communication approach. Each swarm will have a Master UAV till a way-point is reached. Then, the master UAV sends them the new coordinates for the next way-point. This process for is repeated for every way-point till the mission is completed.

All the components mentioned above work together in the UAV to successfully fulfill a mission. The GPS sends the coordinates to the flight controller to make it aware of its current relative position. The flight controller takes all the relevant information from the GPS, sensors, and messages to identify the next step in the task. The flight controller sends and receive information from the surrounding UAVs or the GCS using the communication module. Fatal failures might occur if any of these components malfunction. For example, an UAV might lose its position in a swarm due to a communication link failure or a fault in the GPS. Such failures can lead to catastrophic results like collisions between UAVs if they don't adhere to the their current location in an UAV swarm formation.

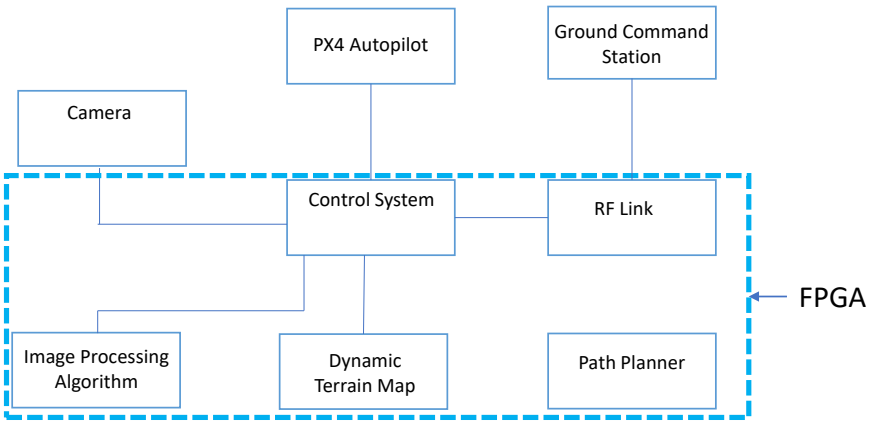

Figure 1.2: FPGA implementation of several UAV components [19].

Components of an UAV like the flight controller and the communication module has been investigated for Field Programmable Gate Array (FPGA)-based implementation by [42]. The overall architecture of an FPGA-based UAV proposed by [19] is shown in Figure.1.2. Different sub-components like the image processing algorithm used to interpret data from the camera and Radio Frequency (RF) link used to communicate with the GCS are constructed and connected together through a control system inside an FPGA. The PX4 Autopilot receive commands from the FPGA to move the UAV accordingly.

Building UAV components using SRAM-based FPGAs potentially makes them easier to adopt because of the technology's flexibility when implementing a design in hardware [26]. This also lowers the barrier of entry which can further reduce the price of an UAV and allow a designer to explore more efficient design alternatives.

1.3 Dependability and analysis techniques

1.3.1 Dependability

This is a term used to encapsulate different attributes of a system. In general, these metrics assess if a system can provide accurate and stable services according to the user's specifications [4]. In our work, we focus on the following dependability attributes:

- **Reliability:** This metric is used to examine the probability that a component of a system will execute the task correctly for a predefined amount of time under specific conditions [13]. For example, during a mission in an UAV, it is crucial to have a navigation and control systems that works for the entire task period [41].

- **Availability:** This attribute is utilized to determine if a component of a system is correctly working when needed, provided that the other external components it depends on functions as intended [41]. For instance, the communication module of an UAV needs to be functioning properly at any instance of time to send and receive messages to other UAVs or the GCS provided that it can receive sufficient power from the battery.

1.3.2 Analysis techniques

UAVs are complex devices composed of many sub-system that work together to complete a mission. Therefore, the research surrounding them is still thriving in various fields. This subsection aims to highlight analysis techniques used to evaluate the dependability of these devices.

- **Simulation:** The aim of this approach is to evaluate the dependability attributes by imitating real world systems [24]. Replicating the behavior of a system in a real world scenario is done through sampling of data over time. The AEROSTACK and ArduSim are two examples of an UAV simulation environment. AEROSTACK is an open source software framework proposed by [47] in 2016. Through their simulation environment, a designer can test the reliability of their UAV in different mission scenarios like visual recognition of an object in an environment and rescue operations in various settings. The simulator however, is complex and expensive in terms of the necessary resources required to run every possible scenario for a mission. The authors of the MUSCOP protocol [18] tested the communication reliability between the master and slave UAVs in different swarm formations using the ArduSim simulator. Their work only focused on packet loss ratio and coordinates offset in a swarm formation based on the distance between the master and slave UAV and the quality of a communication channel. However, other vital conditions, such as

the transmission power, application environment, or the message transmission delay, were not considered.

- **Formal approaches:** This technique analyze a system by first modelling it mathematically or through a Finite State Machine (FSM). Then, a property of interest is asserted to verify if the system can satisfy it or not. A formal approach does not require any stimulus or sampling to verify a property. Hence, it can verify a property of a system quickly, with less computational resources, and provide trustworthy results [6]. In the following paragraphs, we focus on several UAV formal analysis frameworks and discuss the limitations of each framework.

  The authors in [25], presents a formal resiliency evaluation framework using satisfiability modulo theory (SMT) to find out the vulnerability of a UAV network if a number of UAVs (k) became unavailable or went missing from the network. The user inputs resource constrained parameters and the framework verifies if this network resilient. A network is said to be resilient if the rest of the UAV network can preserve the communication between every UAV and the central navigation UAV either directly or indirectly through message hopping. Some limitations are however present in this work. First, the effects of a communication component is only fatal if the overall network becomes unavailable due to a missing UAV. Hence, the point-to-point communication failure was not considered. Additionally, SMT can only provide a true of false analysis based on the asserted property without any probability of failure or success. This can be limiting in terms of parameter calibration to attain a specific dependability threshold.

  Utilizing symbolic, simulation, and statistical model checking methods, the authors in [34] introduced a formal framework to calculate the number of successfully reached destinations by an UAV. Through their framework, the user can find the number of UAVs that returns to the base station and visited points of interest based on parameters such as wind chance and caution levels due to lower battery life. The work presented in this paper did not consider the impact of a communication component on the mission quality.

  The reliability of an UAV system was investigated by [5] using a parametric statistical model checking framework. Using their framework, a designer first

build a parametric model of an UAV system. Then, evaluate the effect of parameters like the wind force and possible failure in a sensor on the reliability of their model. In their paper, the authors focuses on the UAV flight controller as a case study. This work only examine failures that occur in a system due to external factors like wind or internal components like filters of an UAV.

1.4 Radiation effect and mitigation

The radiation effect on hardware components was first noticed in the 1980s. Errors in electronic components caused due to radiation became more notable in the 1990s and early 2000s. This motivated the scientific community to analyze this behavior carefully and introduce several solutions to mitigate these faults. The following subsections provide a brief overview of radiation effects, their causes, the two most common techniques used to mitigate these errors , and relevant work in the literature regarding radiation vulnerability and mitigation analysis techniques.

1.4.1 Radiation effect

An error can manifest and propagate if a radiation particle hits sensitive areas like the transistor of a device. These errors can affect the state of the transistor temporarily or even permanently in some cases. Decreasing the node technology of a device allows us to squeeze more transistors in the same circuit area. However, this makes the radiation effect more prominent and increases the chance of component failure due to radiation-induced errors [15]. SRAM-based FPGAs suffer even more from such errors compared to other electronic components. This is mainly due to the massive area occupied by SRAM memory which is more prone to radiation-induced faults in a device [3]. These errors can have major implications on a system and, therefore, needs to be mitigated to make the components of a system dependable in a radiation environment.

1.4.2 Radiation Mitigation

The two most common approaches used to mitigate radiation effect in a device are the blind scrubbing and the Triple Modular redundancy (TMR) techniques. The

blind scrubbing technique re-configures an FPGA from a golden copy periodically to ensure that any error manifested from radiation is resolved. Two fundamental parameters are configured when designing a blind scrubbing component. The first parameter is the period between two scrubs (i.e., scrubbing interval) depending on the expected radiation rate [1]. Also, it is necessary to set up the time to re-configure the FPGA (i.e., scrub time) based on the size of the implemented design.

The TMR approach relies on using multiple identical blocks that perform the same computational task. The result from each component is then fed as inputs to a voting component. Then, the final output from the voter is chosen based on the most repeated result from these identical blocks [50].

Compared to TMR, blind scrubbing is a more suitable mitigation technique in an UAV application for two main reasons. First, it draws less power and, hence can save more battery life. Also, it has a smaller design footprint, making it more beneficial in the UAV's tightly constrained area.

1.4.3 Vulnerability and Mitigation Analysis Techniques

The authors in [23], introduced MAVFFI to evaluate the effects of errors such as soft or hard errors on the reliability of a UAV. Their approach looks into the end-end reliability of the UAV processor rather than providing a stage-by-stage analysis. Using this technique, only faults that can propagate and affect the overall reliability of a UAV can be addressed through mitigation. They also proposed an anomaly detection and recovery scheme using software to mitigate radiation-induced faults. Their framework, however, looks only at the processor used for an UAV disregarding any other vital components like the communication module. Also, their recovery scheme mainly relies on software techniques, so hardware mitigation techniques were not considered.

Hardware susceptibility and fault mitigation techniques for soft errors were previously explored in different research domains such as low orbit satellites. In [2], using probabilistic model checking (PMC) and continuous-time Markov modeling, the authors presented means to perform system-level vulnerability analysis on the LEON3 data path. They also offered models to find the optimal self-repair rate for a processor under SEUs. Continuous Markov chain and PMC were also used by [22] to

measure the dependability of an FPGA device in an aerospace setting. Their models concentrated on different error repair scenarios under various constraints such as performance, area, and dependability for each environment.

1.5 Problem Statement

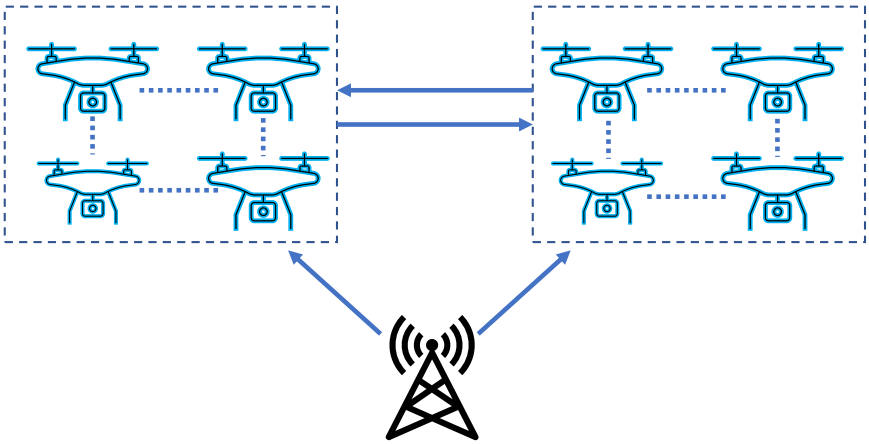

Figure 1.3: UAV communication network [20].

Ideally, UAVs can communicate together in a swarm, with the GCS, or even other UAV swarms in their vicinity to exchange information and receive new missions. An example of this ideal network is shown in Figure.1.3 where UAVs can transfer messages between each other. Additionally, they can establish a communication link with the other UAV swarm in the network. Moreover, the GCS can communicate with any cluster of UAVs in its area [20]. Several questions come to mind when discussing this ideal network for UAVs built using SRAM-FPGAs in a radiation environment:

- What telemetry device should we use ?
- How can we measure the point-to-point reliability of the communication link between UAVs based on the chosen telemetry device specifications ?
- In a radiation environment what is the impact of using the blind scrubbing mitigation technique on the availability of the communication modules ?
- What is the optimum scrubbing interval for the communication components ?

From our literature survey, we noticed that the dependability of an UAV was evaluated based on the sensors used or the impact of environmental factors like wind on them. However, the effect of FPGA-based UAV communication components on the link between UAVs or the GCS has not been considered in a radiation environment. Thus, in this thesis, we propose a methodology to evaluate the reliability and availability of an UAV based on the telemetry device and communication protocol in a harsh radiation environment.

1.6 Thesis Contribution

The work presented in this thesis aims to evaluate and measure the reliability and availability of a communication protocol through a novel probabilistic timed automata (PTA) framework using different parameters depending on the environment and component constraints. Based on the aforementioned discussion in the problem statement, the contribution of this thesis can be summarized as follows:

- Build models to represent conditions that can impact the UAV-UAV communication. The framework utilizes variables such as the transmission power, antenna gain, UAV movement speed, SEU rate, scrub time, number of messages, transmission delay, receiver sensitivity threshold, and telemetry device specification to estimate the reliability of a communication protocol. We target these parameters because they differ based on the targeted UAV design and application environment and therefore, are expensive to re-evaluate every change that occurs in them through simulation. The The MAVlink replacement negotiation scenario is used as a case study communication protocol for our framework because it is built on the MAVlink V1.0 protocol which is an open-source and well established communication protocol use for UAV-GCS communication.

- Extending the framework by removing abstractions in the calculation of SEU rate and implementation of the blind scrubbing technique. This is achieved by designing and implementing the Register Transfer Level (RTL) code of serial communication components to extract the number of configuration bits and calculate the number of potential critical bits. Also, implementing the time between scrubs to trigger the scrubbing behavior. Increasing or decreasing by a huge margin might affect the availability of the communication protocol. Thus,

we identify the optimum scrubbing interval for a UAV-UAV communication component through our framework.

- Expanding the framework to include UAV to GCS communication under a soft fault caused due to radiation. Additionally, different kinds of faults are injected in the communication between GCS and UAV when a message is transmitted from the UAV. This allows us to assess the effect of various failures caused by soft fault and measure the limitations of blind scrubbing for such failures. Then, we measure the link availability when using the blind scrubbing mitigation technique. Finally, we show the viability of our methodology by comparing them to similar experiments in the literature.

1.7 Thesis Outline

The rest of this thesis is organized as follows:

- In chapter 2, we present the preliminary associated with our framework followed by a brief introduction to model checking and the UPPAAL-SMC tool.
- In Chapter 3, we first discuss our overall methodology. Then we highlight our PTA models for the UAV-UAV and UAV-GCS frameworks. Finally, we present the PTA used for our case-study communication protocols.
- In chapter 4, we provide our results for the different test scenarios and discuss the effect of various parameters on the reliability and availability of the case study communication protocols.
- A summary of our work in this thesis and future areas of interest in research are discussed in chapter 5.

Chapter 2

Preliminaries

The preliminaries required for our work are presented in this chapter. We begin with discussing single event effect on FPGAs and scrubbing as a mitigation technique followed by the case study communication protocols used in our work. We then provide a brief explanation about Probabilistic Model checking, semantics, and the UPPAAL-SMC tool. Finally, we end this chapter by presenting a summary for the chapter.

2.1 Radiation Effect On Hardware components

Several types of errors can manifest due to radiation, but the most prevalent type is SEE errors. SEE faults occur when a single highly energetic particle hits a sensitive circuit component causing a malfunction in the device [30]. Errors formed due to SEE in a circuit can be classified into two main categories [16]:

- **Soft errors:** Soft errors are born from radiation events that create a charge to overturn the state of circuit elements like flip-flops or latches. These faults are soft because the hardware element is not permanently damaged, as the device can save new data properly. Therefore, the glitches are momentary and non-destructive. Soft errors can be further classified into four types put as shown in Table.1.

Table 1: Types of soft error [38].

Soft error types	
Type	**Characterization**
SEU	Single event Upsets (SEUs) errors are caused when a radiation particle strikes an individual storage location.
SEL	Single event effects caused due to anomalous high-current states that triggers a parasitic dual bipolar circuit are known as Single Event Latchup (SEL) malfunction.
SET	Temporary bugs occuring at a single node that transits in a combitional logical circuit and can be captured by a storage component, are known as Single Event Transient (SET) faults.
SEMT	Single Event Multiple Transient (SEMT) are similar to SETs but rather than having a glitch at a single node, glitchs occur at multiple nodes.

- **Hard errors:** Unlike soft errors, hard errors are nonrecoverable and destructive to a circuit. SEE causes permanent changes in circuit elements such as flip flops or gates. Hard errors can be classified as shown in table.2.

Table 2: Types of hard error [38].

Hard error types	
Type	**Characterization**
SEB	Permanent damage to a circuit due to high circuit states are called Single Event Burnout (SEB).
SEGR	The gate oxide of the circuit elements is damaged, hence, the resulting path becomes corrupt. These errors are called Single Event Gate Rupture (SEGRs).

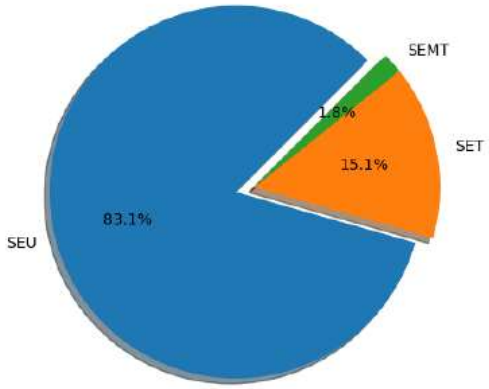

Figure 2.1: SEE research [38].

Based on Figure.2.1 from [38], we observe that SEU dominants most of the research for SEE. This indicates that SEU needs to be addressed for any component in a radiation environment. Hence, in our thesis, we focus on the effect of SEUs regarding UAV-UAV and UAV-GCS communication. Failures from SEUs can be distinctly noticed in avionics and aerospace applications but can be found in any circuit in any environment with high radiation activity. In some environments, UAVs can be susceptible to such malfunctions. For example, a group of UAVs can be operating in a natural disaster area where the radiation levels are high or relay critical radiation information back to the control station.

2.2 Scrubbing

Scrubbing is the process of periodically inspecting the device to solve any potential faults caused by soft errors. This approach requires fewer elements making it more simple and requires less overhead compared to other fault rectification techniques like triple modular redundancy (TMR) [46]. Also, it can repair errors quickly because the frequency of the scrubbing can be modified depending on the application. This error correction technique can be implemented by having only a correction algorithm or adding a detection method to read back memory and scrub only frames affected by SEUs. Solely using a correction algorithm is known as blind scrubbing. The technique above uses a non-volatile storage element on-chip to save an unaffected version known as a golden copy. A timer configured by the designer is used to know when scrubbing

should occur. Once the set time is up, the scrubbing process starts, and the device resets. This process makes the device return to regular operation in case there is an SEU or MEU [22].

Components handling scrubbing can be implemented on or off-chip. On-chip blind scrubbers are cheaper because it does not need an additional processor to perform the scrubbing process. While On-chip scrubbing components are less expensive, they can be susceptible to faults, and therefore, in some applications, an external radiation-hardened component is required to maintain the availability of the device. In power and area-constrained application such as an UAV, internal scrubbing is a more applicable implementation. The rate at which scrubbing occurs and the expected upset of an UAV component directly impact the device. Therefore, in our work, we implement and measure the effect of using the blind scrubbing technique on the availability of a communication protocol component.

2.3 MAVlink protocol

The MAVlink protocol is one of the most popular UAVs and GCS communication protocols currently used [28]. It was first introduced (v1) as a bidirectional high-level header-only communication protocol for unmanned systems in 2009 under the LGPL licenses by Lorenz Meier. Then, the developer released an updated version in 2017. This new version (v2) adds additional bytes and security fields to the protocol and maintains backward compatible with the v1 of the protocol.

The MAVlink protocol has been adopted in different applications like agriculture, and environmental inspections [36][48]. The protocol is community-driven and well-established, where the community can add additional messages. The following subsections discusses the frame of a MAVlink message and an extension proposed by [14] to enable UAV to UAV communication in swarms.

2.3.1 MAVLink protocol frame

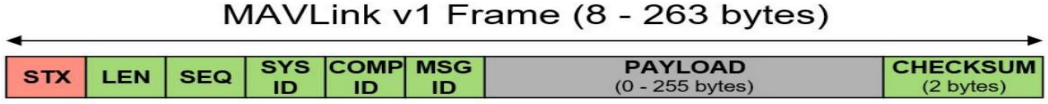

Figure 2.2: MAVLink V1.0 frame [28].

Messages sent between the UAV and GCS are binary serialized messages where the content is sent from the transmitter, and the sequence is reconstructed in the receiver. The MAVlink protocol message frame can be anywhere from 8 bytes and 263 bytes depending on the message type, and the payload sent as shown in Figure 2.2. A message starts with a constant byte (0XFE), followed by the payload's length and the message's sequence. In V1.0 the system field size is only 1 byte. Hence, there can only be 254 UAVs used with system 255 reserved for the GCS. The MAVlink protocol supports 27 different hardware types specified by the component (COMP ID) field. A message sent through the MAVlink protocol in the message ID (MSG ID) field is categorized as a status or command message. Status messages are sent from the UAV to the GCS to update the location, velocity, and provide updates on the status of different components. An UAV is informed of its next task from the GCS through a command message. The developers provided additional MSG IDs for the community to add more messages types. This provides more flexibility to the user and increases the functionality and robustness of the protocol over time. The content of a message content is found in the payload field. The checksum field is used to confirm that no alterations occurred in a message from a malicious attack.

The MAVlink protocol relies on telemetry communication devices like Bluetooth or network interface devices like WIFI to deliver messages. However, messages can only be exchanged between a single UAV and the GCS at any given instance, leading to a singular failure point if any problems occur in the GCS. Additionally, this singular point of communication does not allow for UAVs to form swarms and communicate with each other independently.

2.3.2 MAVLink Extension

The extension for the MAVlink protocol [14] built on V1.0 focuses on three main aspects regarding UAV communication namely, the group formation, swarm formation, and swarm maintenance. The authors presented, six command messages, 33 messages and 6 type enumerations are as additional message types to the MAVlink protocol to achieve UAV communication. To maintain compatibility with the MAVlink protocol, any message that an UAV can send with updates for the group, swarm, or replacement messages, can also be sent from the GCS. In the following paragraphs, the characteristics of this extension will be discussed briefly.

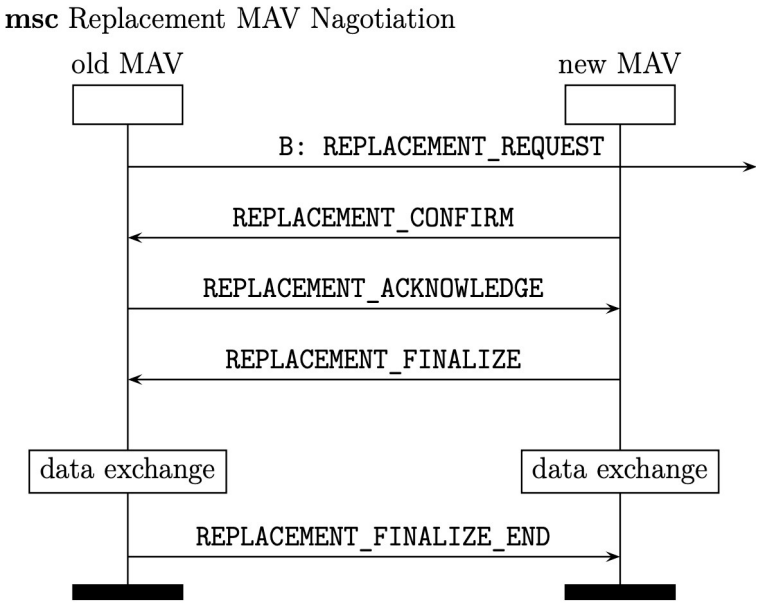

Figure 2.3: Replacement negotiation scenario [14].

- **Group formation :** A group is initialized by sending a group formation message to nearby UAVs, providing them with two essential fields, the group ID and group member ID, with the UAV creating the group being the UAV group leader. The group leader is responsible for broadcasting messages like a new UAV joining the group or appointing a new leader to all UAVs in the group. Alternatively, the GCS can also be responsible for group formation and leadership assignment with the group request messages. The GCS sends a message to each UAV containing its group ID and group member ID. The maximum number of UAVs in a group is 255 groups. Two unique group IDs numbers are reserved. Group ID 0 addresses all UAVs, and group ID 254 addresses UAVs without groups.

- **Swarm formation :** Is defined as a group of UAVs performing the same task. After creating a by an UAV or a GCS, the group can then initiate a swarm formation. The authors describe 4 types of swarms namely, the circle, grid, V, and X swarm formations.

- **Swarm maintenance :** Swarm maintenance replaces an UAV due to either battery exhaustion or faults with another UAV to continue a task. For the UAV replacement protocol, Figure.2.3 discusses the necessary messages required to

accomplish the replacement. The UAV needing replacement broadcasts UAV replacement requests to nearby available UAVs without tasks. The replacing UAV confirms the request. After receiving a request, the exhausted UAV sends back an acknowledgment message. The replacing UAV then sends a replacement finalizing message to initiate the data exchange between the two UAVs. Successfully exchanging data between the UAVs then sends an end message to allow the replacing UAV to take the place of the replaced UAV. The replaced UAV then is relieved of its task to go for maintenance if there is a fault or to a battery charging station if the battery needs recharging. The replacement negotiation scenario aims to solve the battery constraints by allowing an UAV to be replaced by another UAV to maintain the swarm formation and allow for an extended flight time to perform additional tasks.

Based on the literature survey, no validation or experiments were performed on this extension. Furthermore, the UAV-GCS communication component availability under the effect of radiation was not discussed. Therefore, the replacement negotiation scenario and the MAVlink protocol are used as case studies for our framework in this thesis.

2.4 Model Checking

Model-checking is an analytical approach used to verify a property in a system formally. It was first introduced independently by Clarke and Emerson and by Queille and Sifakis in the early 1980s [10]. Compared to other rigorous approaches like simulation, formal checking excels in two key areas. First, it is relatively fast in producing results. This is mainly because these techniques rely on looking at a system's state space rather than sampling points, which is tremendously less expensive in terms of computation and time. Moreover, formal methods are very exhaustive techniques compared to simulation [9]. This is achieved by looking at all the possible scenarios or paths that can lead to a particular state. A serious challenge that affects typical mathematical model checking techniques sufferers is scalability [32]. Classic model checking approaches find it difficult to implement complex systems as they only work for special systems with particular structures. Several techniques were proposed to evade this obstacle. One of those techniques is to assert a property using stochastic

analysis on a network of PTAs. By performing hypothesis testing for a finite set of experiments, statistical evidence can solve properties on a more extensive system. The following subsection discusses the UPPAAL-SMC, a well-established tool used to perform SMC on networks of PTA.

2.4.1 UPPAAL-SMC

UPPAAL-SMC is an extension of the UPPAAL tool that uses stochastic timed automata (STA) to efficiently perform statistical model checking (SMC) on model properties. For model formalism, UPPAAL-SMC uses an extension of the timed automata (TA) to achieve PTA. This technique provides a probabilistic replacement for non-deterministic choices in enabled transitions and probability distribution refinements for non-deterministic time delays [11]. The tool allows the designer to use probabilities both for the input and the outputs of a system. This is notable compared to other formal strategies that can only provide a true or false output. Also, this approach works with tolerance levels rather than absolute values. These features make it popular for communication protocols [8]. In our thesis, we use the UPPAAL-SMC tool to build our PTA framework. Additionally, the query language made to verify a property is utilized to measure the reliability and availability of the communication protocol. The following paragraphs provide simple examples of a PTA model consisting of two states implemented in the UPPAAL-SMC tool and Queries to highlight the features of the tool:

Invaria

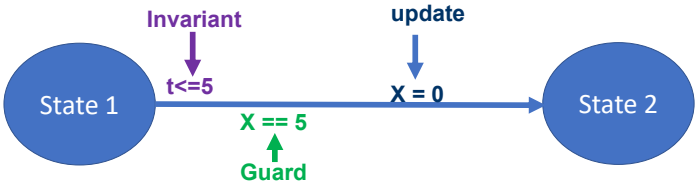

Figure 2.4: PTA example 1

The first example model shown in Figure 2.4 is composed of two states, the *State1* state and *State2*. Each state in the model has three main attributes. A time-invariant indicated by the color purple is used to force the system to leave a state when the

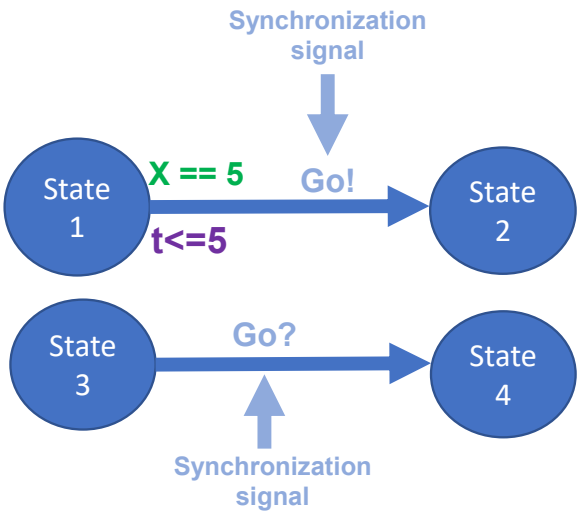

Figure 2.5: PTA example 2

Synchronization creates a channel between different models to trigger transitions between them through signals. Using the signal *GO* presented in Figure 2.5 a channel is formed to link the transition from *State 1* to *State 2* and *State 3* to *State 4*. Once the invariant and guard of *State 1* are satisfied, the model transitions to *State 2*, at the same time *State 3* moves to *State 4*. The signal *GO?* waits for the initiation of *GO!* signal to trigger the move from *State 3* to *State 4*.

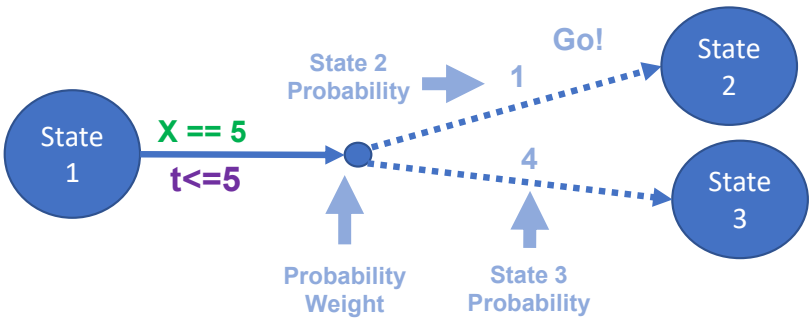

Figure 2.6: PTA example 3

The *weighted probability* state shown in Figure 2.6 is used to provide a probabilistic choice when moving between states. After satisfying the guard and time invariant in *State 1*, the model model moves to the *weighted probability*. The chances of reaching *State 2* and *State 3* are $\frac{1}{5}$ and $\frac{4}{5}$. The PTA takes the *State 2* path if the $\frac{1}{5}$ probability is taken. Otherwise it moves to *State 3*.

Properties and Queries

The UPPAAL-SMC tool uses weighted metric temporal logic queries to verify the property of a state in a network of PTAs. Through the use of queries, three types of verification queries can be performed for any state [8]:

1. **Hypothesis testing:** Investigates the probability of a state being higher than or equal to a certain threshold. The threshold can be any value between 0 and 1. For example, **Pr[≤100](<>test.sample1) ≥ 0.5** checks whether the unknown probability of the *sample1* is greater than or equal to 0.5.

2. **Probability evaluation:** Produces an approximate interval of the probability of reaching a state in the PTA network. An example of this query is **Pr[≤100](<>test.sample2)** looks at the probability of reaching the *sample2* within 100 time units.

3. **Probability comparison:** Used to evaluate whether the probability of reaching one state is greater than the probability of reaching the other state. For

instance, **Pr[≤100](<>test.sample1 ≥ Pr[≤100](<>test.sample2)** determines the which probability is larger within 100 time units.

In our thesis we use probability evaluation queries for our framework. We present three different queries to measure the reliability and availability for the UAV-UAV and UAV-GCS communication protocols.

2.5 Summary

In this chapter, we presented all the necessary preliminaries to understand our proposed framework. We briefly discussed the radiation effect on hardware components and why addressing SEUs is crucial in harsh environments for an UAV. We also explained and show why scrubbing is a more viable option than TMR for small UAVs. We then provided a concise overview of the MAVlink communication protocol, the replacement negotiation scenario, and our motivation to use these protocols. Finally, we briefly discussed model checking and summarized the main features of the UPPAAL-SMC tool used in this thesis.

Chapter 3

PTA Framework Methodology

3.1 Introduction

This chapter presents a detailed description of each PTA model in our framework for both UAV-UAV and UAV-GCS protocol communication protocol frameworks. Initially, the overall methodology is presented. Then, the proposed framework for the UAV-UAV communication protocol reliability is discussed. After that, improvements to the UAV-UAV framework model for SEU rate calculation and communication PTAs are presented. These improvements are utilized to measure the availability of such protocols under radiation. Finally, we put forward the different PTA models used to build the UAV-GCS availability analysis framework and the PTAs for the case study communication protocols .

3.2 Methodology Overview

The work presented in this thesis utilizes Probabilistic timed automata to build a novel framework that can evaluate the reliability and availability of a communication protocol based on the used communication components and the radiation environment. Fig3.1 shows an overview of our methodology. Initially, the RTL code for the communication protocol components used in the UAV is extracted to calculate the SEU rate using the number of potential critical bits and failure rate of a bit in a deployment device (λ_{bit}). The Xilinx Vivado tool is used to find the number of potential critical bits by synthesizing and implementing the RTL code for a target device.

After calculating the SEU rate, the communication protocol is modeled as a as Finite State Machine (FSM) and implemented as a PTA template. The generated PTA template of the communication protocol is placed as the top-level model in either the UAV-UAV or UAV-GCS framework. Then crucial parameters such as the maximum UAV speed, scrub interval, and message transmission delay are inserted along with the SEU rate to our framework. Finally using the UPPAAL-SMC tool and property

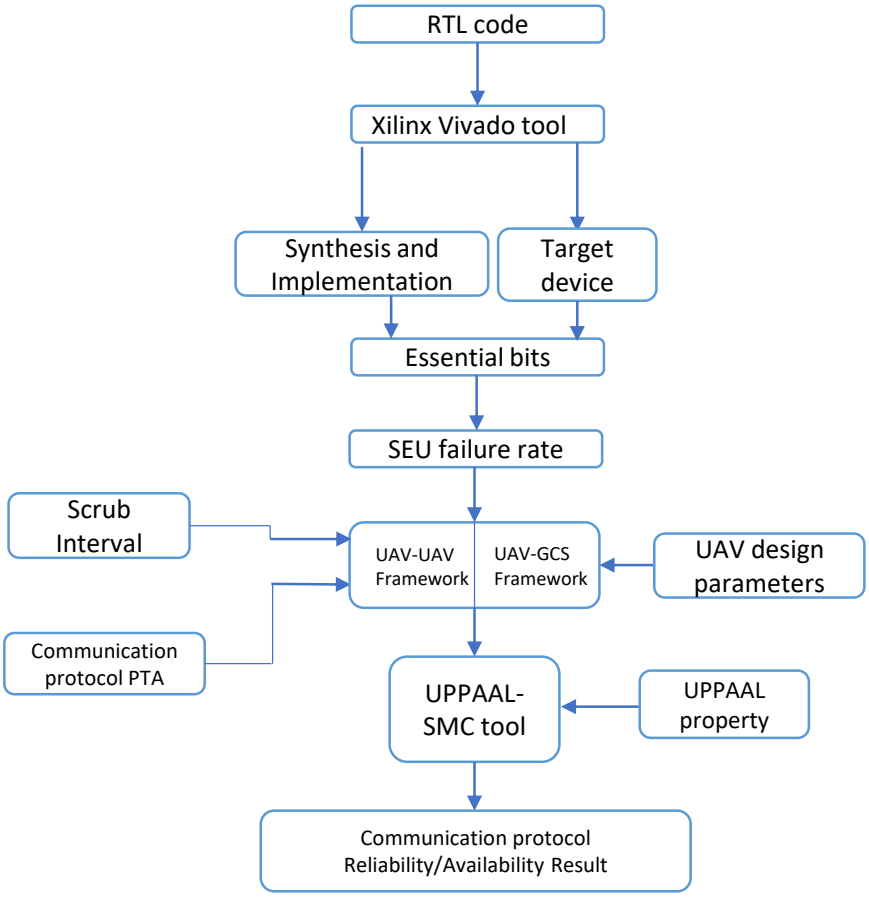

Figure 3.1: Overall Proposed Methodology.

3.3 UAV-UAV framework

This framework deals with communication protocols involved with direct communication between UAVs. Several factors can affect the reliability of such devices. Some

of these factors can be from the UAV itself, such as battery levels which can directly affect the transmission power and sensitivity of the communication module, and the transmission delay before sending a message. Other factors can be external to the UAV, like radiation environment exposure. Therefore, we build six PTA models to emulate the speed of an UAV and have the current coordinates in a 3D plane, the status of the transmitter, and the receiver in the event of SEU. Our framework estimates the reliability of these protocols by looking into parameters such as the transmission power, antenna gain, UAV movement speed, SEU rate, scrub time, number of messages, transmission delay, receiver sensitivity threshold, and telemetry device specification parameters to estimate the reliability. These variables depend on the expected power consumption and targeted environment. A communication failure occurs if the signal strength falls below the sensitivity threshold or the transmitter or receiver models are unavailable due to an SEU. The following subsections describe the equations and the PTA models used to build this part of the framework.

3.3.1 Free Space Propagation Equation

At high altitudes and in clear weather, UAVs have unobstructed lines of sight. Hence, the communication between them is only limited to signal strength. We first compute the distance between two UAVs (U1 and U2) using the Euclidean Equation to calculate the received signal strength. Where (x_1,y_1,z_1) represents the coordinates of U1 (U_1) and (x_2,y_2,z_2) represents the coordinates of U2 (U_2) in the x,y, and z space respectively as shown in Equation (1) [40]. Additionally, we use Equation (2) to determine a telemetry device's wavelength (λ). This Equation utilizes the speed of light (c) and frequency of the telemetry device (f) to find λ. Finally, Equation (3) computes the received signal strength in a fair-weather condition [43]. In this equation P_t stands for the transmitted power, G_t is the transmitter gain, G_r is the receiver gain, λ is the wavelength, d is the distance between the two UAVs, L is the system loss factor not relating to the communication, and P_r is the received signal's strength.

$$d(U1, U2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \tag{1}$$

$$\lambda = \frac{c}{f} \tag{2}$$

$$P_r = \frac{P_t G_t G_r \lambda^2}{(4\pi)^2 d^2 L} \tag{3}$$

3.3.2 UAV Movement PTA model

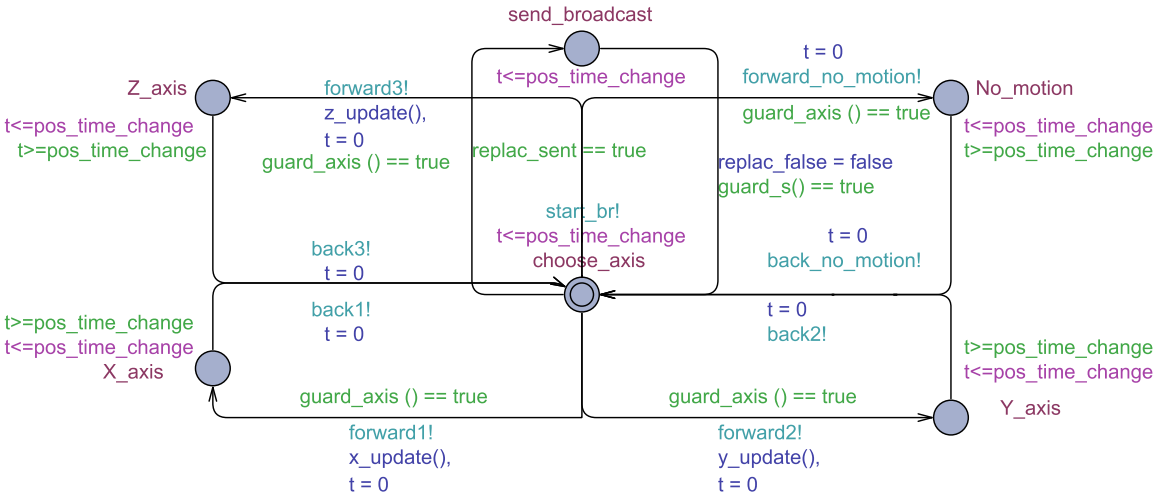

Figure 3.2: Proposed UAV movement PTA model.

The PTA model presented in Fig.3.2 aims to express the free motion of U1 in air. Our model starts from the *choose_axis*, a broadcast message is triggered by setting the *start_br* signal to "true" and transitioning to the *send_broadcast* state. This PTA launches the replacement negotiation scenario between U1 and U2 when *start_br* is triggered. Following the initiation of the protocol, the PTA moves back to *choose_axis* state and waits till the *pos_time_change* invariant is satisfied. By fulfilling this invariant, the PTA can move to one of four states. The movement and stabilization time required by U1 is abstracted by using the *pos_time_change* invariant. Each axis is presented by a single state, hence, the *x-axis*, *y-axis*, and *z-axis* states emulate the UAV movement in the air. After moving to one of these states, the *x_update* function is called to update the airspeed. The change in airspeed corresponds to a change in the coordinates of U1. This new airspeed is a random integer between 0 and *UAV_speed*. The variable *UAV_speed* is initialized before starting the experiments. Randomizing airspeed aims to provide a high-level abstraction for the UAV movement speed by considering negatively impacting elements like variable acceleration and air resistance. The *No_motion* state emulates no movement in the UAV as it stays

in its current coordinates. Except for the transition that initially occurs from the *choose_axis* state to *send_broadcast*, all the other states are assumed to carry equal probability weights.

3.3.3 Transmitter PTA model

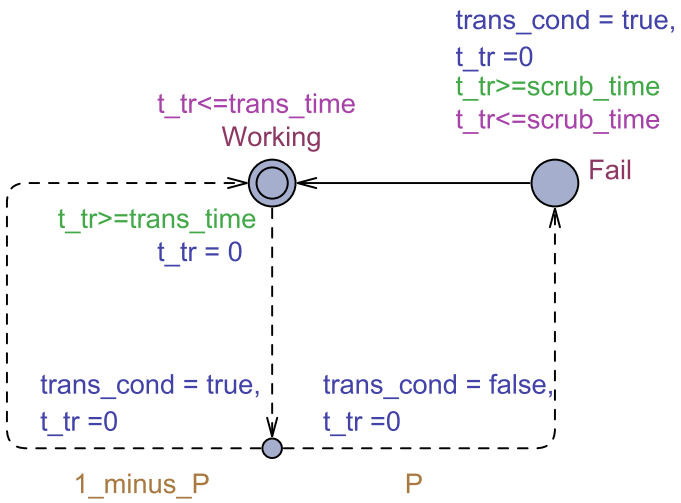

Figure 3.3: Proposed UAV transmitter PTA model.

The transmitter PTA shown in Fig.3.3 aims to encapsulate the behavior of the transmission phase in a communication module that uses the blind scrubbing technique to mitigate SEUs. To represent this behaviour, the model is composed of the *Working* state, *Fail* state, and the *weighted probability* state . Initially, our model starts from the *Working* state, and the transmitter is assumed to be functioning properly. This is indicated by the *trans_cond* variable being *"true"*. The PTA then proceeds to the probability weight state after the *t_tr* time-invariant is satisfied. In this model, *P* refers to the SEU rate of the transmitter, and *1-P* is the probability that an SEU fault will not occur. The model returns to the *Working* state if it takes the *1-P* and *trans_cond* remains true . Otherwise, if the *P* leads to the *Fail* state, the *trans_cond* becomes *"false"*. Changing this variable demonstrates a faulty change in the functionality of the transmitter and that a message cannot be transmitted when needed. By fulfilling the *" $t \geq$ scrub_time"*, the transmitter returns to an error-free by changing the *trans_cond* back to *"true"* and the PTA transitions to the *Working* state.

3.3.4 Receiver PTA model

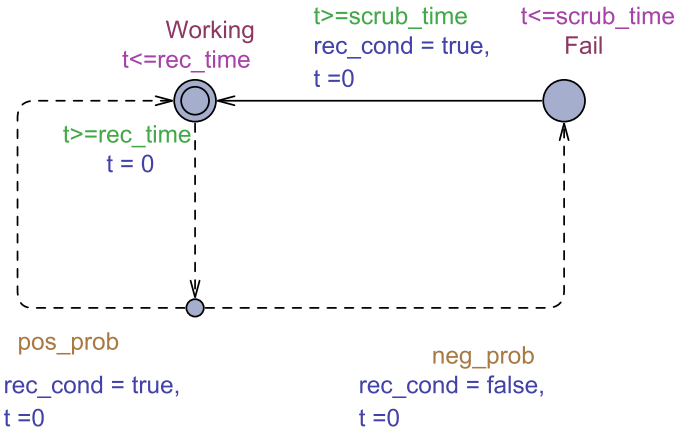

Figure 3.4: Proposed Receiver PTA model.

Similar to the transmitter PTA, the receiver model presented in Fig.3.4 aims to represent the effect of blind scrubbing for the serial receiver component under the presence of SEUs. The *pos_prob* and *neg_prob* variables are utilized to show the availability and failure rate of the receiver. We use the *rec_cond* to indicate the current condition of the receiver. When the *"t ≥ rec_time"* guard is concluded the model moves to the *weighted probability* state to randomly choose the next state of the receiver. The value of *rec_cond* goes from "true" to "false" if the model takes the *neg_prob* path. Scrubbing is then initiated using the *"t ≥ scrub_time"* and the *rec_cond* variable becomes "true" when the PTA returns to the *Working* state.

3.3.5 Data Exchange PTA model

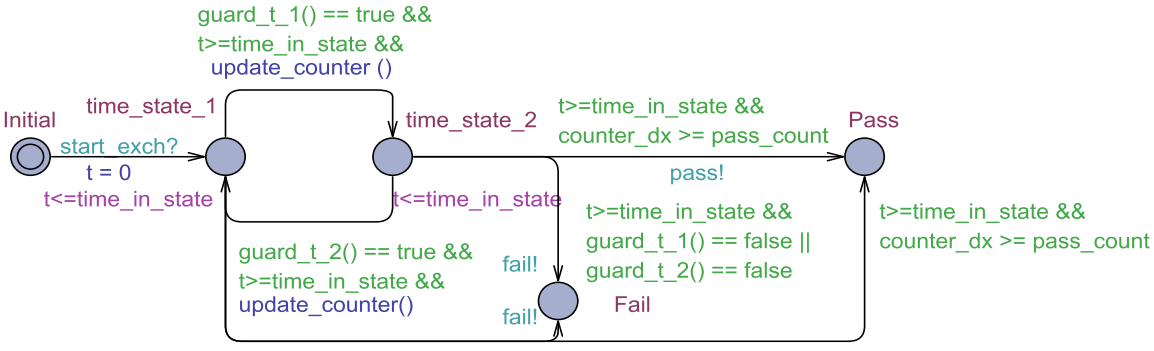

Figure 3.5: Proposed UAV-UAV data exchange PTA model.

The data exchange model in Fig3.5 is initiated after receiving the *Start_exch* signal from the communication protocol. In our work, it is triggered by the *Replacement_Finalized_received state* from the replacement negotiation scenario PTA (Fig.3.12). When this signal is received, the model goes to the *time_state_1* state. Four main conditions are necessary to move from the *time_state_1* state to the *time_state_2*. First, the PTA confirms that the number of messages needed to conclude is still not yet reached through the "*counter_dx* ¡ pass_count" guard. Additionally, we implement the duration for the message to leave the transmitter (i.e., Transmission Delay) using the "t $\geq$ time_in_state". Also, the PTA decides whether a message is sent successfully by checking the condition of the transmitter and receiver using the *trans_cond* and *rec_cond*. Finally, the computed received signal strength must be greater than or equal to the specified threshold of the communication device. If all these conditions are true, the model transitions to the *time_state_2* and the *update_counter* is called to increment the number of messages successfully sent by one. Through these guards, we implement the conditions necessary for U1 to send a message to U2 correctly.

The *time_state_2* state implements all the aforementioned guards. If all the guards are satisfied, the model moves back to *time_state_1* and the *counter_dx* is increased by one. These conditions indicate that U2 has successfully sent a message back to U1. Returning the model to the *time_state_1* creates a loop between the two states to emulate the data exchange between two UAVs. Once all the required messages are received, the guard "*counter_dx* $\geq$ pass_count" is satisfied and the model transitions to the *pass* state. The *pass* signal is set when the model transition to this state. If any message cannot be delivered either due to the availability of the transmitter and receiver or the power not being enough for the receiver to sense it, the model moves to the *Fail* state and sends the *fail* signal back to the top-level communication PTA.

3.4 Extending the UAV-UAV framework

Several abstractions were present in the section mentioned above regarding SEUs and the implementation of the blind scrubbing technique. First, the SEU rate applied in the framework was a general value taken from [37]. Thus, it does not provide a proper estimate for the SEU effect on an UAV telemetry device. Besides using a standard SEU rate, the transmitter and receiver models only considered the time it

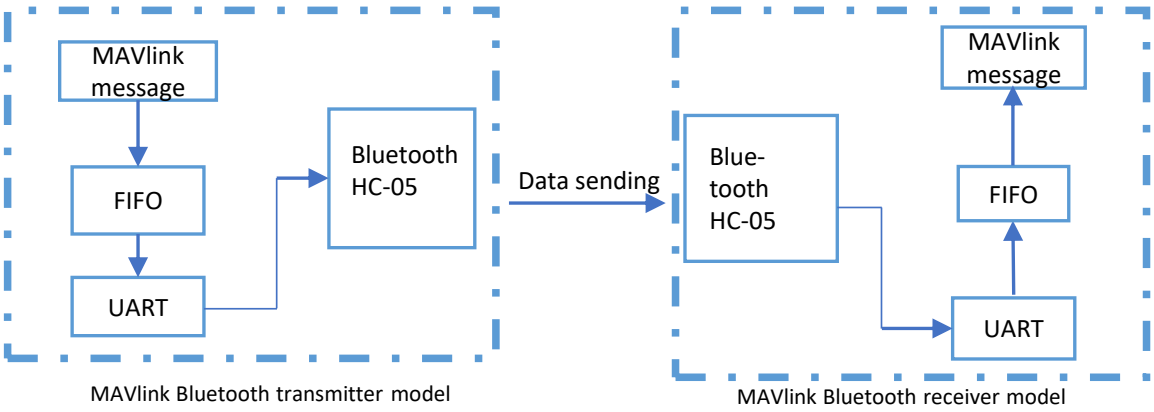

Figure 3.6: Serial transmitter and receiver design.

For a serial communication protocol, the transmitting UAV must store and convert a message into bits before sending it to the telemetry module in a serial communication protocol. Also, the received bits need to be reconstructed and stored as bytes correctly in the receiver buffer of the second UAV. The implementation of the MAVlink transmitter and receiver models in an FPGA is shown in Figure 3.6. First, we design a FIFO to store the maximum bytes for a MAVlink message (8-263 bytes). When it is time to send a message, the Universal Asynchronous Receiver Transmitter (UART) converts bytes into bits and transmits the message. The UART reconstructs the bits into bytes on the receiver side, and the MAVlink message is then stored in a FIFO for later use. The Bluetooth HC-05 is shown as an example of a telemetry device connected to the UART. The following equation is used to calculate the first-order worst-case scenario SEU rate for the message construction and deconstruction components in the UAV:

$$\lambda_{device} = \lambda_{bit} \times Number\ of\ Critical\ Bits \tag{4}$$

In Equation (4), λ_{bit} refers to the rate at which a bit flip occurs for a finite time interval. The *Number Of Critical Bits* signifies the number of bits sensitive to SEUs. We use the λ_{bit} and *Number Of Critical Bits* variables to estimate the failure rate of

a device (λ_{device}). To find the *Number Of Critical Bits* for the serial communication component, we first implement the design as an RTL code. Then, we use the Xilinx Vivado "*generate bit stream*" feature to extract the *Number of critical bits* needed for the FPGA device. The *Number of critical bits* bits obtained for our design using this tool is 62335. The λ_{device} can be more accurately analyzed using techniques such as fault injection as shown in [27]. Nevertheless, we can estimate and use the first-order worst-case of an SEU rate using the *Number of critical bits* obtained from the tool as shown in [21].

3.4.2 Improved transmitter PTA

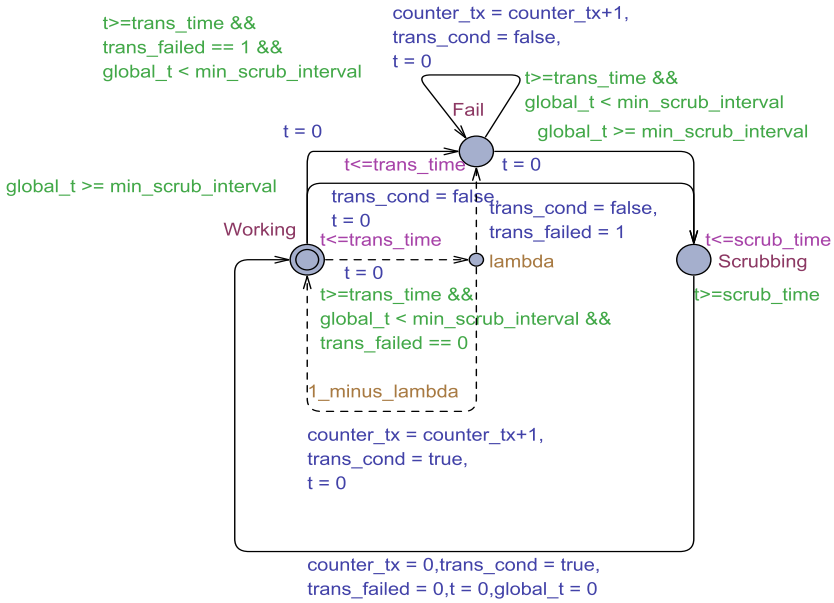

Figure 3.7: Proposed extension to the transmitter PTA.

The new transmitter PTA presented in Figure 3.7 consists of four states instead of three states. Similar to the original transmitter PTA our model starts from the *Working* state and the *trans_cond* variable remains "true" in this state. After satisfying the *trans_time* guard, the model moves to the *weighted probability* state. The availability and failure rate of the transmitter (*1-P* and *P*) in the original transmitter PTA is also utilized here. Through the *weighted probability* state the transmitter can go to the *Fail* state if the *P* path is taken or return to the *Working* state if the *1-P* path is chosen. If the model moves to the *Fail*, the *trans_cond* variable becomes "false"

to indicate that the transmitter is unavailable and cannot send data to other UAVs. The model remains in this *Fail* state until the "$global\_t \geq min\_scrubbing\_interval$" guard is fulfilled. This guard is implemented to mimic the scrubbing interval behavior needed to trigger the scrubbing of the device. The variable *global_t* is the clock used for the transmitter, and *min_scrubbing_interval* is a variable initialized before the beginning of the experiment that signifies the necessary scrubbing interval required between every scrub for the transmitter. The *trans_cond* is updated or remains false, and the model transitions to the *Scrubbing* state immediately after this guard is satisfied regardless of its current position. This state is used to represent the time required for the transmitter to reconfigure itself and remove the SEU present in it. By clearing the *"scrub_time"* guard, the scrubbing process is completed, and the PTA returns to its working condition. This is implemented by having the PTA transition to the *Working* state and updating the *trans_cond* from "false" to "true".

3.4.3 Improved Receiver PTA

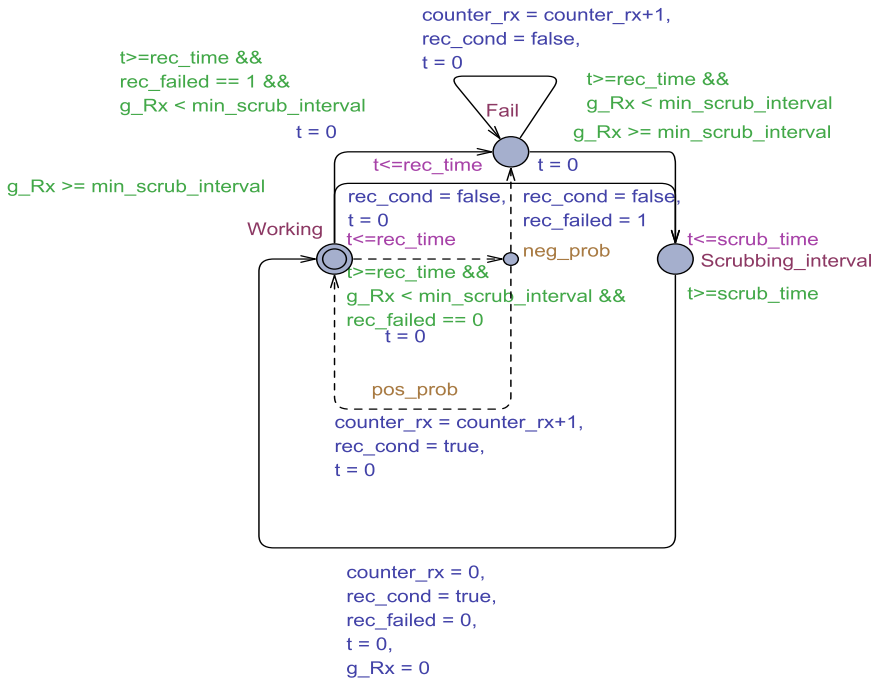

Figure 3.8: Proposed extension to the receiver PTA.

The receiver PTA shown in Figure 3.8 adheres to the same design of the transmitter PTA but with several modifications in the clocks and variables used. First, the *g_rx*

clock replaces the *global_t* clock to avoid any racing conditions when scrubbing is needed. Additionally, the model uses the *rec_cond* variable to represent the condition of the receiver. The *rec_cond* becomes "false" if a fault occurs or when the transitions to the *Scrubbing* state occurs. After scrubbing the receiver, the model moves back to the *Working* state. Going back to the *Working* state updates the *rec_cond* back to "true" and the *g_rx* clock is reinitialized for the next scrub.

3.5 UAV-GCS framework

We further expand our overall methodology through the UAV-GCS availability analysis framework. In this framework, we look at the effect of blind scrubbing on the UAV-GCS communication under SEUs. The fault injection, blind scrubbing, and UAV ground speed PTAs are constructed to measure the availability of such protocols. In the following subsections, we fully define the behavior of each PTA and present the equations used to calculate the ground speed of an UAV.

3.5.1 Fault Injected PTA model

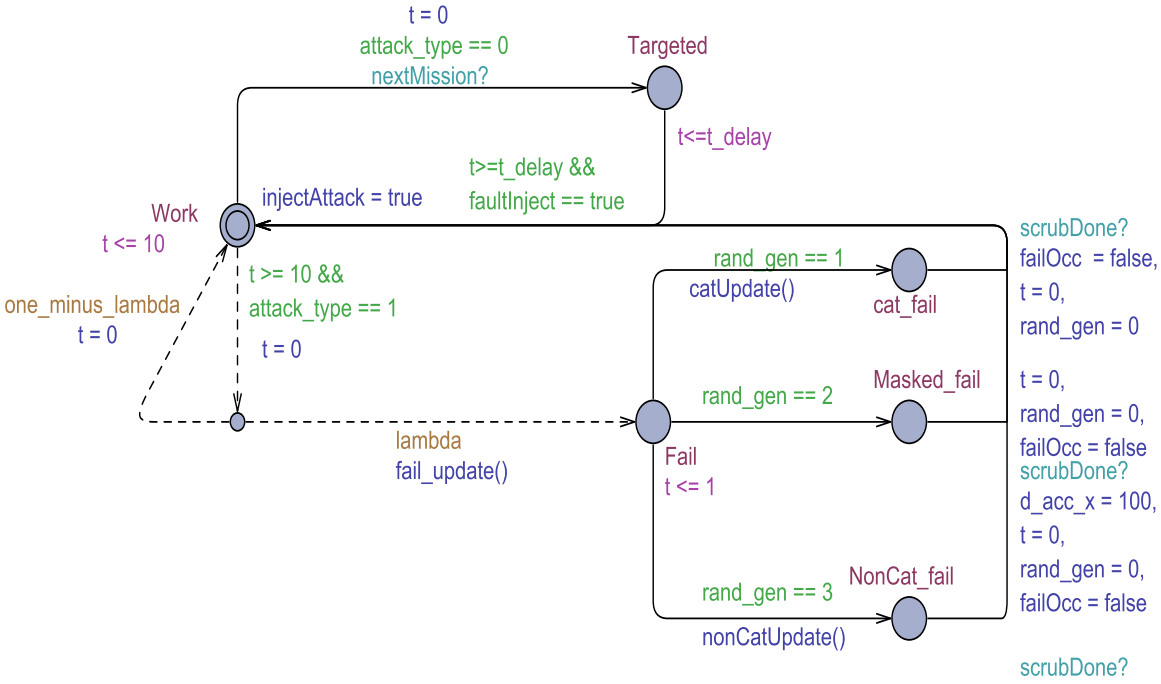

Figure 3.9: Proposed fault injection PTA model.

SEU faults can be injected either directly or randomly for a communicating UAV through the PTA shown in Figure 3.9. Before starting an experiment, the type of fault injection is chosen by the *attack_type* variable. The fault caused due to of such SEUs is classified into three main categories:

- **Catastrophic:** These faults are detrimental to the communication. If they occur, a message cannot be transmitted or received in the UAV .
- **Non-catastrophic:** Are silent fault that does not affect the link between the UAV and GCS but can have sever complication in the success of the overall mission. An example of such faults is a change in the content of a message before sending it from the UAV.
- **Masked:** The SEU fault is concealed and hence, do not cause any problem to the UAV.

If the user sets *attack_type* to zero, the PTA becomes geared towards direct fault injection for a specific state in the UAV PTA. In this work, fault injection occurs when the GCS sends a new mission request to the UAV. The *nextMission* signal is sent to make the PTA wait till the last mission is transmitted to the UAV PTA before moving to the *targeted* state from the *Work* state. The fault is injected into the UAV PTA (Figure 3.15) once the last mission request is received from the GCS PTA (Figure 3.14). This is indicated by updating the *injectAttack* variable to "true" after the guard *"faultInject == true"* is fulfilled.

Random faults can be injected when the *attack_type* to one. In this mode, the model transitions from the *Work* to the *weighted probability* state when the guard "t $\geq$ 10" is satisfied. Similar to the *weighted probability* state in Figure 3.7, the model can take either the work or fail path using the variables *1-P* and *P*. If the work state path is chosen, the UAV message transmission continues to be successful. However, if the *P* path is selected, the PTA moves to the *Fail* state to randomly select one SEU fault through the *rand_gen* variable in the *fail_update* function. A catastrophic fault occurs if the model transition to the *Cat_fail*. The *catUpdate* function is used to update the *failOcc* to "true" indicating that the UAV can no longer send or receive messages. For a non-catastrophic fault to occur *rand_gen* must return 3 from the *fail_update*. We assume that any non-catastrophic fault leads to a change in the movement speed

of the UAV to highlight the severe complications that can happen caused by such faults. In this state *nonCatUpdate* function is used to change the speed of the UAV through the variable when the PTA transition the *NonCat_fail* state. Finally, for the *masked* state faults, no change occurs in the system. Therefore, no variables are modified in this state. In our work, we assume that all three fault types can occur with carrying equal probabilities. Setting the *scrubDone* signal in blind scrubbing PTA (Figure 3.10) resolves any fault caused due to SEUs, and the model returns to the *Work* state.

3.5.2 Blind Scrubbing PTA model

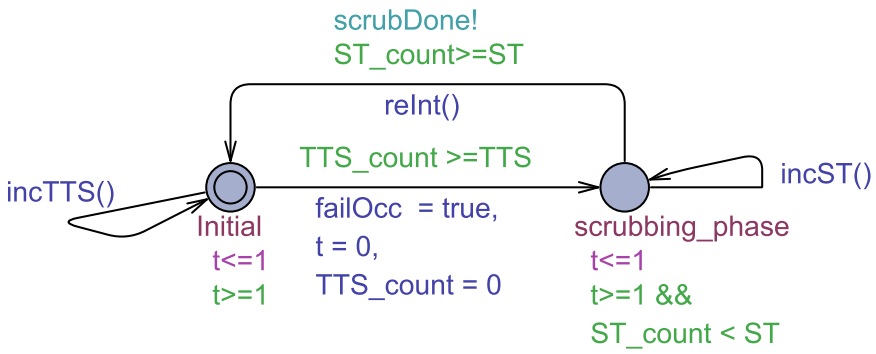

Figure 3.10: Proposed blind scrubbing PTA model.

The implementation of the blind scrubbing technique for the UAV communication component is presented in Figure 3.10. Starting at the *Initial* state, the model checks if the scrubbing process can start using the "$TTS\_count \geq TTS$". In this guard, TTS stands for the scrubbing interval and TT_count represent time since the last scrub. The $incTTS$ function is called every one time unit to reset the clock and increment the TTS_count variable by one. To initiate the reconfiguration of the FPGA, the model transitions to the *scrubbing_phase* when TTS_count must be equal to or exceed the predefined scrubbing interval. The PTA waits for the '$ST\_count \geq ST$" to be satisfied, where ST_count is the time currently spent on the process, and ST is the interval of time required to conclude the procedure. By returning to the *Initial* state, the signal *scrubDone* is set to drive the fault injection PTA back to the *Working* state. This transition signifies that the transmission link is now available, and messages can be exchanged between the UAV and GCS. Finally, the *reInt* update function is

utilized to prepare the PTA for the next scrub by resetting all necessary variables like *ST_count*.

3.5.3 Speed PTA model

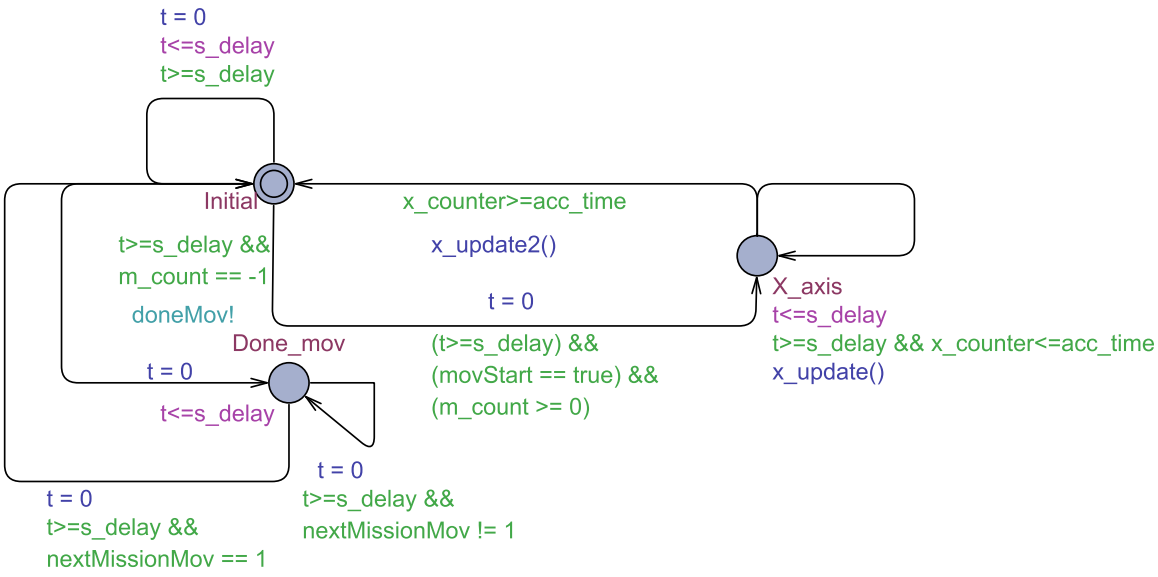

Figure 3.11: Ground Speed PTA model.

The PTA presented in the Figure 3.11 is used to implement and compute the ground speed of an UAV in the x-plane. Ground speed (Gs) is the speed of an object relative to the surface of the earth in the presence of wind [45]. At first, the model waits till all the necessary messages for each mission are communicated. It is assumed that all these messages refer to movement in the x-axis. Once all of them are received, the PTA is notified using the *movStart* variable. When it becomes "true", the motion of the UAV is initiated by moving to the *x_axis* state. This state aims to imitate the motion of an UAV in the air. Every communicated message is used to signify the acceleration of the UAV for a certain distance. This behavior is implemented through the guard "$x\_counter \leq acc\_time$", where *x_counter* counts the number of steps taken, and *acc_time* is the total number of movements required to conclude the message. Additionally, the guard is used to "$t \geq s\_delay$" to signify the time taken to complete a single motion in the air.

The *x_update* function is called to increase the speed of the UAV given that the "$t \geq s\_delay$" and "$x\_counter \leq acc\_time$" conditions are satisfied. The increase in

the speed is realized by the following equation [45]:

$$Gs = As\text{-}Wx \tag{5}$$

In Equation (5), As represents the UAV speed in air and Wx is the x-component of the wind speed. Subtracting Wx from As gives us the Gs in the x-axis. We use Equation (6) uses the wind speed (W) and its direction (θ) to calculate Wx [33] :

$$Wx = W \cos\theta \tag{6}$$

The *x_update* function is also responsible for incriminating the *x_counter* by one and resetting other variables such as the clock when called. When all the steps required for a message are finished, the model returns to the *Initial* state, and the *m_count* variable is decreased by one. This variable is used to amount for the number of messages in a mission. If all the messages are concluded, the PTA transitions to the *Done_mov*. The model remains in this state until the end of the final mission. Otherwise, it returns to the *Initial* state to prepare for the next mission.

3.6 Case study PTA

3.6.1 Replacement Negotiation Scenario

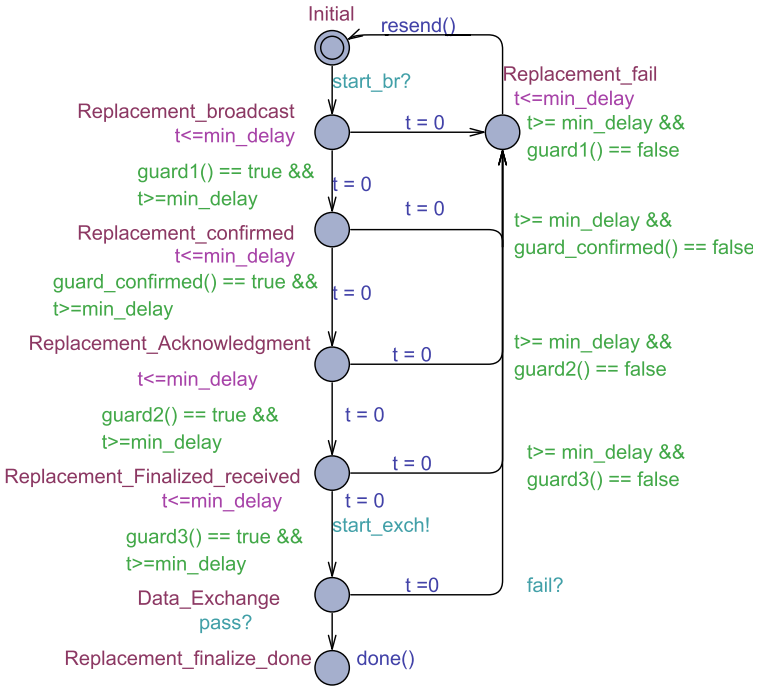

Figure 3.12: Proposed Replacement Negotiation Scenario PTA model.

The replacement negotiation scenario discussed in Fig.2.3 is implemented as a PTA in Figure 3.12. The model stays in the *Initial* state until the *start_br* signal is set by U1 (UAV that needs replacement). A link needs to be established before exchanging any data between U1 and U2. To form this connection, the PTA must pass through the *Replacement_broadcast*, *Replacement_confirmed*, *Replacement_Acknowledgment*, and *Replacement_Finalized_received* states. Each of these states represents a message transmitted between U1 and U2. Similar to the data exchange PTA, this model looks at two necessary factors to send a message correctly. First, the received signal strength is calculated and compared to the sensitivity threshold of the device. Along with the transmitter and receiver must both be available (i.e., *trans_cond* and *rec_cond* must be "true"). Transitioning to the next state occurs if all of these conditions are true. However, if the received signal strength is not enough or the communication component is not available, the message fails to be transmitted, and the PTA moves to the *Replacement_fail* state. The link is finally established when the model transitions from the *Replacement_Finalized_received* state to the. This transition triggers the *start_exch* signal to initiate the data exchange between the UAVs. The data exchange PTA returns the *Pass* signal if it is successfully concluded. Otherwise, the *fail* signal is raised to indicate that the communication between the UAVs has failed. The model then transitions to the *Replacement_fail* state. U1 can re-initiate the replacement negotiation scenario if the *Replacement_fail* is reached.

3.6.2 MAVlink communication Flow

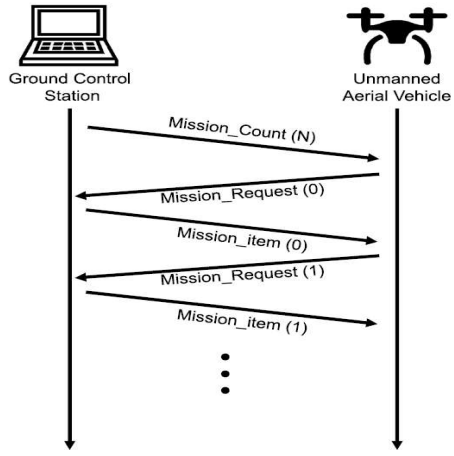

Figure 3.13: MAVlink communication flow [29].

The MAVlink communication flow shown in Figure 3.13 dictates how the UAV and GCS interact with each other [29]. Initially, the GCS sends a message that communicates the number of messages (*Mission_count*) required from the UAV. The UAV reciprocates with a request for the first mission, and the GCS returns the content of the mission using the *Mission_item* message. This process is repeated until the UAV receives all the missions. After getting all the necessary messages, the UAV proceeds to perform its tasks. The UAV and GCS PTAs designed to follow the MAVlink communication flow are discussed in the subsequent paragraphs.

GCS PTA model

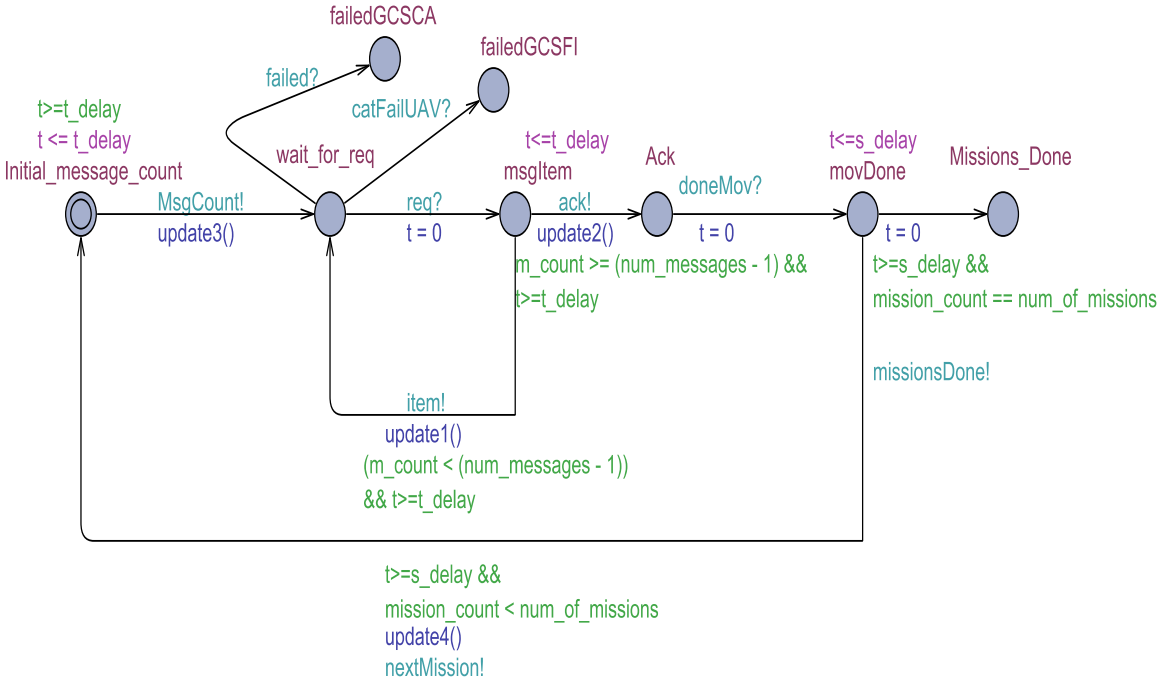

Figure 3.14: Proposed GCS PTA model.

The PTA shown in Figure 3.14 executes the essential elements for the GCS. Initially, the model starts at the *Initial_message_count* state. A transition from the *Initial_message_count* to the *wait_req* state occurs after the "$t \geq t\_delay$" guard is satisfied. This transition sets the *MsgCount* signal to indicate that the UAV have received *Mission_count* and can start sending requests. The PTA stays in the *wait_for_req* state until it obtains a request from the UAV PTA through the *req* signal. Once this signal is triggered, the GCS can start sending mission item messages from the *msgItem*

state. The GCS returns to the *wait_for_req* after transmitting this message to wait for the next request. This return to the *wait_for_req* creates a loop between the UAV and GCS. The loop ends when the model transmits the last mission item from the GCS to the UAV. The item messages are complete by exchanging this final message, and the model moves to the *Ack* state. In this state, the *movStart* becomes "true" to initiate the motion missions assigned to the UAV through the speed PTA (Figure 3.11). The model goes to the *movDone* state after successfully finishing all of the items and checks whether there are any more missions. If this was the last mission, the model transitions to the *Missions_Done* state. Otherwise, it goes back to the *Initial_message_count* to send the *Mission_count* message. The model can go to either the *failedUAVCA* or *failedUAVFI* states depending on the type of fault injection specified by the user. Further elaboration around injected faults will be present in the UAV PTA model.

UAV PTA model

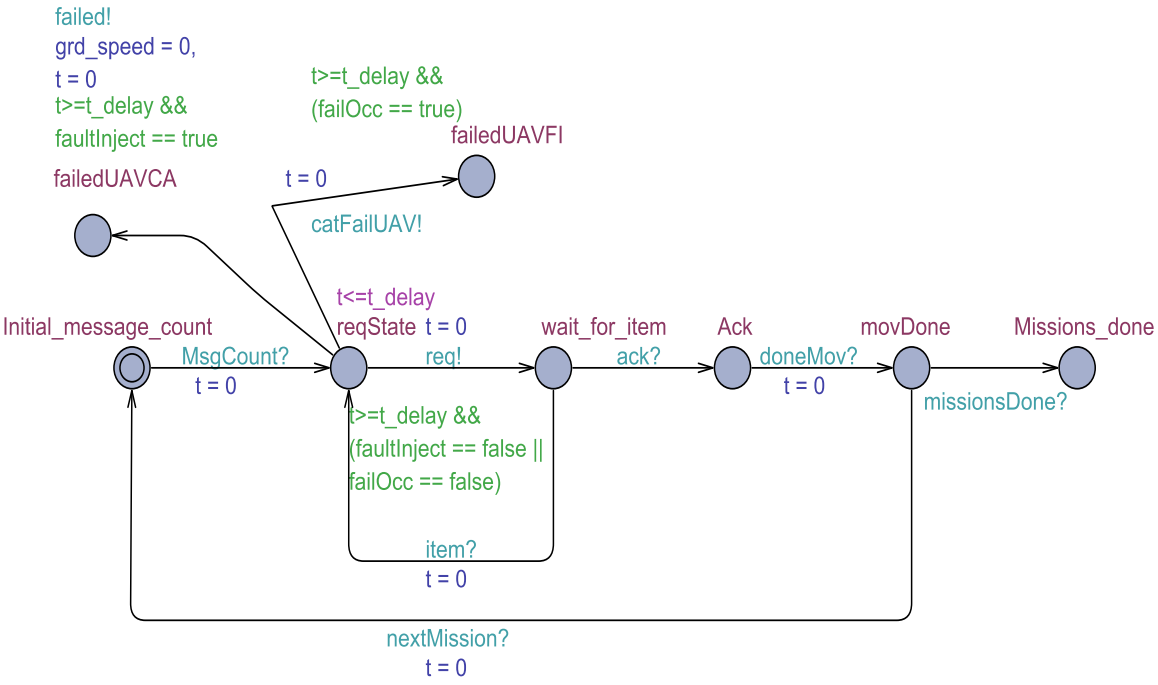

Figure 3.15: Proposed UAV PTA model.

Implementation of the UAV MAVlink communication flow is shown in Figure 3.15. This figure discusses the signals, states, and the effect of catastrophic failure on

the communication of the UAV. After receiving the *Msg_Count* signal, the UAV is ready to send mission item requests from the *reqState* state by setting the *req* signal. The UAV PTA sends all the requested messages correctly when no fault is present. However, the model fails if a catastrophic fault is injected when the UAV is sending this message. Transitioning to the *failedUAVFI* state requires the *faultOcc* variable to be "true". Moving to this state triggers the *catFailUAV* signal for the GCS PTA. Direct faults are assumed to occur when the *MsgCount* of the final mission is sent. It forces the PTA to fail by moving to the *failedUAVCA* state and triggering the *failed* for the GCS PTA.

3.7 Summary

This chapter provided a comprehensive explanation for all the PTAs and equations implemented in our proposed methodology. Initially, we discussed the UAV-UAV framework and how we built each PTA and used the free space propagation equation to evaluate the reliability of a communication protocol. We then highlight some of the initial abstractions present for the SEU rate and the implementation of the blind scrubbing technique for the transmitter and receiver. These abstractions were addressed by extending our framework by calculating the first-order worst-case SEU rate specific to the serial communication component of the UAV through our RTL code design of a UART and FIFO queue. We also extended our transmitter and receiver PTA to account for the time between scrubs (i.e., scrubbing interval). Furthermore, we expanded upon our overall methodology by building another framework to measure the effect of the blind scrubbing technique on the availability of an UAV-GCS communication protocol. Finally, we discuss the implementation of our case study protocols as PTA models. In total, our overall methodology for both the UAV-UAV and UAV-GCS frameworks is composed of 11 PTAs. These PTAs are used to measure the reliability of UAV-UAV communication and the availability of UAV-UAV and UAV-GCS protocols.

Chapter 4

Results and Discussion

4.1 Introduction

This chapter presents our experiments and findings by performing reliability and availability analysis on the case study protocols using our proposed frameworks. First, we perform four experiments to measure the reliability of the replacement negotiation scenario. Next, we conduct six more experiments to evaluate the effect of the blind scrubbing technique on the availability of communication components in the UAV-UAV and UAV-GCS communication protocols. Then, we perform a comparative experiment to highlight the viability of our methodology by comparing to similar experiments in the literature. After that, the reliability and availability experiments are discussed to draw valuable conclusions and have a better understanding of trends that affect the communication of the UAV. Finally, we review these results to learn the current limitations of our methodology.

4.2 Reliability Analysis

The reliability of the replacement negotiation scenario is measured using V1.0 of the UAV-UAV framework. Arudopilot UAVs utilize the MAVlink communication protocol. Therefore, we base some of our specifications on their platform. For the telemetry device, we use Bluetooth, as it provides a decent link range between the UAVs. Also, it draws the least amount of power when compared to other devices like Robsense and SiK radio V2 [28]. The receiver captures any message with a received

signal strength of -80 dBm or higher. Using 2402 MHz for the frequency of telemetry device gives us with a λ of 0.125 through Equation (2). According to their manual, the speed of an UAV can be up to 13 m/s according to their manual [44]. Since U1 is the UAV that needs replacement, it is expected to fly at lower speeds to save battery. Therefore, we set a conservative maximum speed of 4 m/s for U1. As U2 is the replacing UAV, we assume that it has a healthy amount of battery left and can travel at higher speeds. The *UAV_speed* variable is set to 8 m/s for U2. Initially, U1 assumes the coordinates of (10, 10, 10) and (5, 5, 5) for U2 in the (x, y, z) plane. By setting L to one, we assume that no power loss happens in the UAV. Through simulation, the authors in [39] presented antenna gains for an UAV. In the following experiments, we use approximated linear values of this simulated gain as bases for G_t and G_r. The P presented in [37] is used as an initial SEU rate for the transmitter and receiver. We also change P by "$\pm$ 0.3" from the results in [37] to test the effect of varying P on the reliability of the protocol.

Our PTAs are parallelly composed in the UPPAAL-SMC to mimic the U1 and U2 communication for our experiments. Through formal model-checking queries the reliability of the replacement negotiation scenario is verified. All of the tests performed in the following subsections uses the "**Pr[≤999](<>rep.Replacement_fail)**". Through this query, the UPPAAL-SMC tool checks if any path will lead to the *Replacement_fail* within 999 time units. In our experiments, every time unit in the tool is assumed to be 1 second. The verification confidence level for every experiment is 99.5%. This confidence level provides results that are highly accurate in an adequate time frame. Components unrelated to the communication such as sensors and motors are assumed to function perfectly.

4.2.1 Experiment 1: Availability and scrub time

The key objective of this experiment is to estimate the effect of P and scrubbing on the communication reliability. Therefore, we set P_t to 4 dBm and the antenna gains (G_t and G_r) to 3.0. Also, we require 10 messages to be exchanged between the UAVs at a transmission delay of 1 second in the data exchange phase. By initializing these parameters, the system can operate without any other potential bottlenecks in the system. In this experiment, scrub time is incremented by 1 second up from 1 second up to 10 seconds for each P experiment.

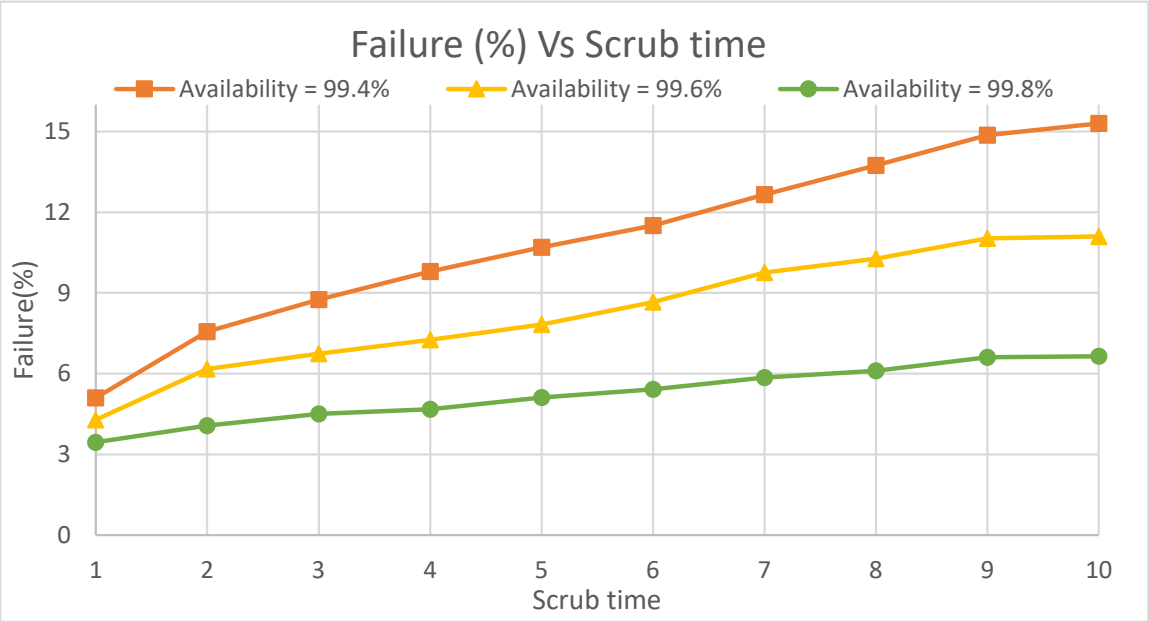

Figure 4.1: Replacement failure at different P values and scrub duration.

The effect of scrub time when P is 0.6%,0.4%, and 0.2% for both the transmitter and receiver is shown in Figure 4.1. Several observations can be seen from this graph. First, the reliability at low scrub times is similar regardless of P. Also, as scrub time increases, the reliability of the overall system decreases. However, the rate of deterioration in the reliability is different for each P. For example, when P is 0.2% and scrub time is 10 seconds, the communication failure is almost 6%. Whereas at 0.6%, the failure is more than 15%. Finally, it is also observed that no mechanism is present to trigger the scrubbing. Due to this observation, the transmitter and receiver models were extended, and further experiments are performed in section.4.3 to address this abstraction.

4.2.2 Experiment 2: Transmitter power and gains

This experiment focuses on the effect of P_t on the model's failure at different gains. Initially, the P_t is set to 2 dBm and incremented by 0.4 dBm up to 4.0 dBm as specified in [28] for the Bluetooth protocol. The experiment is conducted again at 3.0 and 3.2 for G_t and G_r to measure the effect of changing the gain for each P_t value. The number of messages required to complete the replacement successfully is 10, with each message requiring 1 second to be sent. For the transmitter and receiver, P is set

0.2%, and the scrub time is set to 5 seconds. As power is crucial and the reliability extremely sensitive to it, each P_t used in the experiment is verified three times, and the average produced of the three runs is taken to be the model's failure.

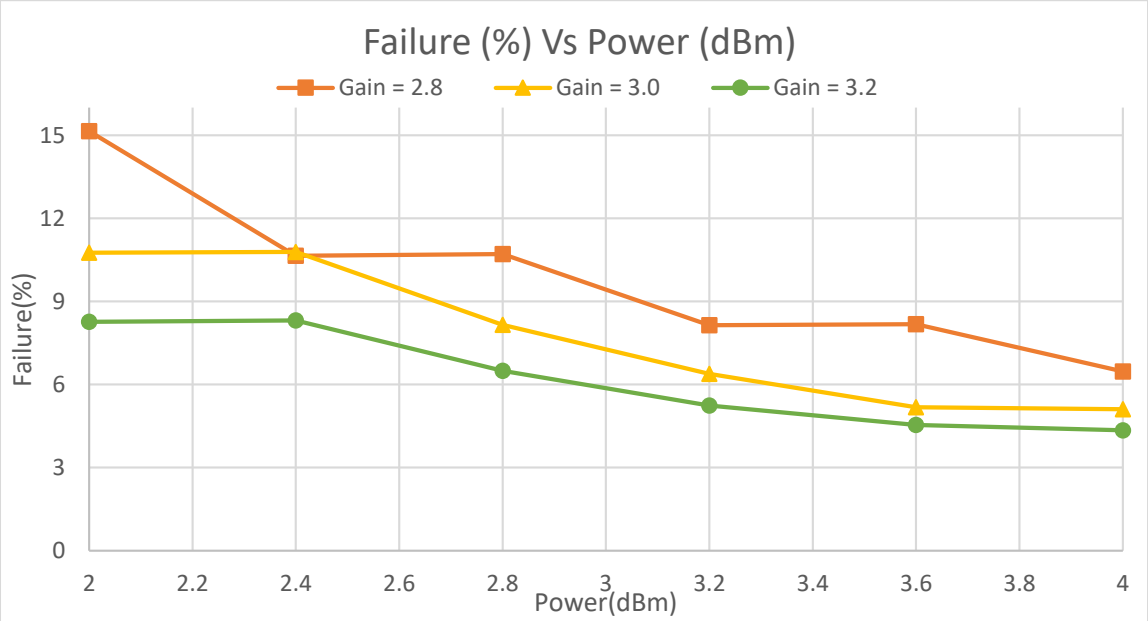

Figure 4.2: Replacement failure at different Pt and antenna gains.

Plotting the results for this experiment in Figure 4.2 indicate that antenna gains have a substantial effect on the communication reliability between U1 and U2 at low P_t. For instance, verifying the protocol when P_t is 2.0 dBm and antenna gain is 2.8. Increasing the antenna gain from 2.8 to 3.2 improves the system's reliability. This is proved by the decrease in the failure rate from 15% to 7% (8% difference).

4.2.3 Experiment 3: Transmission delay and number of messages

Measuring the reliability of the replacement negotiating scenario in this experiment depend on the transmission delay and the number of messages necessary for exchanging data between UAVs. Hence, we increment the number of messages necessary by one for each experiment run in this scenario, starting from 1 message up to 10 messages. The transmission delay for this experiment is initially set to 1 second. Then, we conduct two additional experiments at 2 and 3 seconds. For the other parameters,

P_t is set to 4 dBm, and the antenna gains are initialized to 3.0. Scrubbing the UAV requires 5 seconds, and P is fixed to 0.2.%. Choosing these values limits the effect on the reliability to only the number of messages and transmission delay.

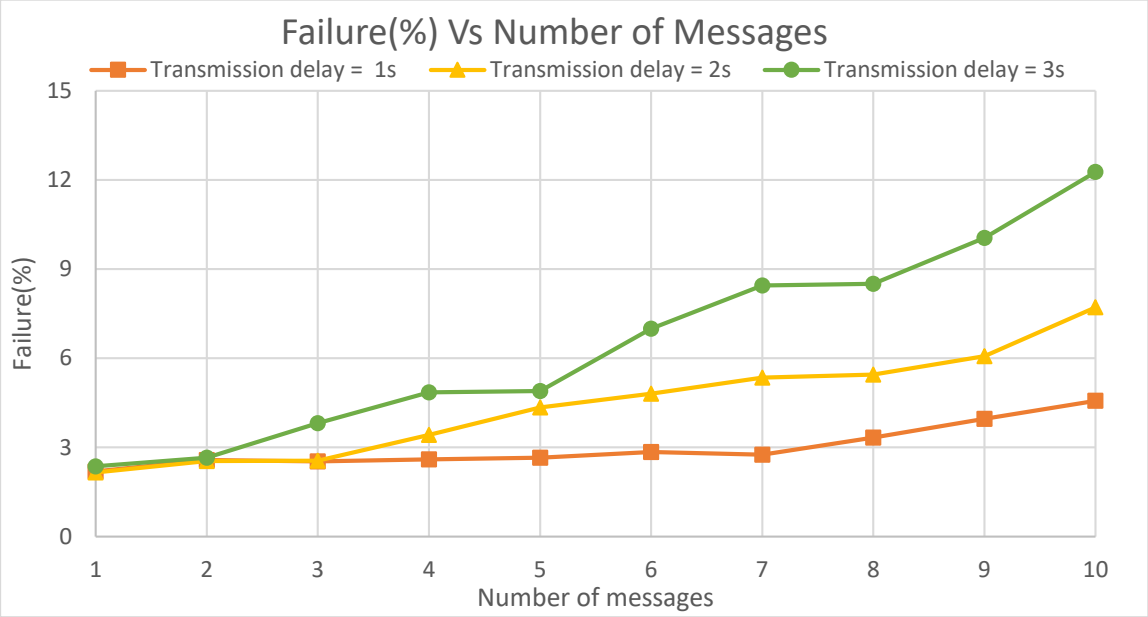

Figure 4.3: Replacement failure at various Transmission delays and number of messages.

Exchanging 2 messages or less has the same effect on the communication reliability regardless of the transmission delay. Increasing the exchanged data beyond 2 messages has a different effect on the reliability based on the transmission delay, as shown in Figure 4.3. The failure always remains below 5% at any given instance if the transmission delay is 1 second. When it is updated from 1 to 2 seconds, the number of messages exchanged within a failure rate of 5% drops to 5 messages. Finally, at 3 seconds, the system can only send 4 messages at this failure rate. The difference in failure rate is almost 7% when 10 messages are exchanged.

4.2.4 Experiment 4: Transmitter power and Availability

In this experiment, we investigate the relation between P_t and P on the message transmission reliability. First, we set the number of exchanged messages to 10 messages. Transmitting a message to the outgoing link requires 1 second. The antennas gains are fixed to 3.0. For the first run, we set P to 99.5% and P_t to 4 dBm. Then,

for each subsequent run P is incremented by 0.1% up to 99.9%. An additional run is performed at 99.99%. The reliability was also investigated at 3 and 2 dBm.

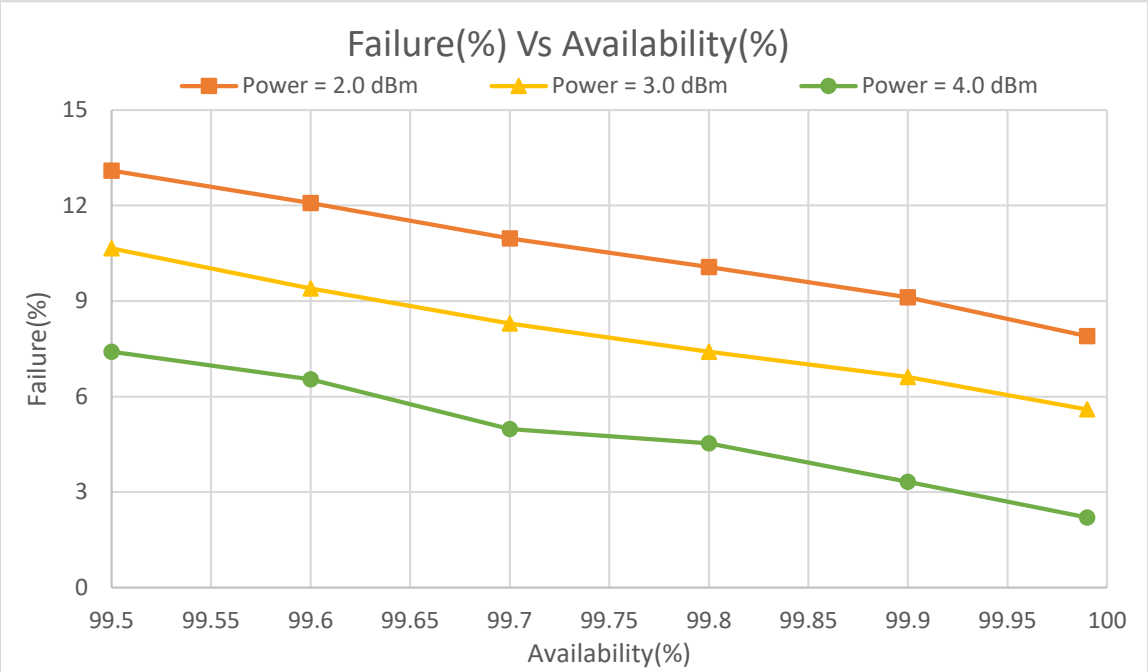

Figure 4.4: Replacement failure at different Pt and antenna gains.

The outcome of these experiments is plotted on Figure 4.4. To achieve 5% or lower failure rate, P mus be less than or equal to 0.3% and P_t must be 4 dBm. Based on these results, both P and P_t equally impact the reliability of the system.

4.3 Availability analysis

Both of our framework's applicability is displayed in the following subsections. These subsections perform different experiments to measure the availability of the UAV-UAV and UAV-GCS protocols using blind scrubbing in the presence of SEUs. First, the effect of the scrubbing interval, number of missions, number of messages, scrub time, and non-catastrophic failures on the availability of the UAV-GCS components is presented in the first four experiments. We run two additional experiments to evaluate the UAV-UAV communication availability. All of these experiments were performed using the calculated SEU rate for Virtex-II and UltraScale Xilinx FPGA products as a case study.

Several assumptions are present when the experiments are conducted. To begin with, we assume that the function of sensors, motors, and other various components of an UAV is correct. Additionally, every second equate to 1 time unit in the tool. Also, only the parameters mentioned above relating to SEUs and blind scrubbing can affect the availability of the UAV-GCS communication. The rest of the communication elements like P_t do not lead to a message drop when the UAV exchanges data with the GCS. Finally, the replacement of UAVs using the UAV-UAV framework occur sequentially, and any SEU fault will cause a catastrophic failure.

Similar to the reliability experiments, the communication stream is simulated by parallelly composing all the models of the frameworks. The query "**Pr[≤99999] (<>GCS.missions_done)**" looks at the probability of successfully arriving at GCS.missions_done for the UAV-GCS framework within 99999 seconds. UPPAAL-SMC verifies if the UAV can successfully receive and accomplish the items provided by the GCS in the presence of SEUs. These experiments are performed at a 99% confidence level. The replacement negotiation scenario's availability is tested using the **Pr[≤99999] (<>rep.Replacement_fail)** query. The tool evaluates the failure probability of the communication protocol within 99999 seconds. The confidence level used for these experiments is 99.6%.

4.3.1 SEU rate for Xilinx devices

The authors in [31] presents the λ_{bit} per day for the UltraScale and Virtex-II devices. We use these values to find the hourly λ_{bit}. Then, the λ_{device} for the devices is calculated using Equation (5) and the number of critical bits discussed in section 3.3. The transmitter and receiver λ_{device} for the case study FPGAs is shown in Table.3.

Table 3: SEU rate for the target device

Configuration Memory bit rate			
Target device	per bit per day [31]	per bit per hour	λ_{device}
Virtex-II	3.99e-07	1.6625e-08	0.0010366
UltraScale	7.56e-09	3.15e-010	0.0000196

4.3.2 Experiment 1: UAV-GCS Availability and scrub interval

This experiment mainly focuses on the availability of the UAV-GCS communication at different scrubbing intervals to find the optimum scrubbing interval for the Virtex-II and Ultrascale devices. Initially, the scrubbing interval is set to 1500 seconds. For each subsequent run it is increased by 100 seconds up to 2000 seconds. We set the $1-\lambda_{device}$ to 99896 and λ_{device} to 104 to emulate the SEU rate of Virtex-II. The λ_{device} is fixed to 2 and $1-\lambda_{device}$ is set to 99998 for the UltraScale experiment.

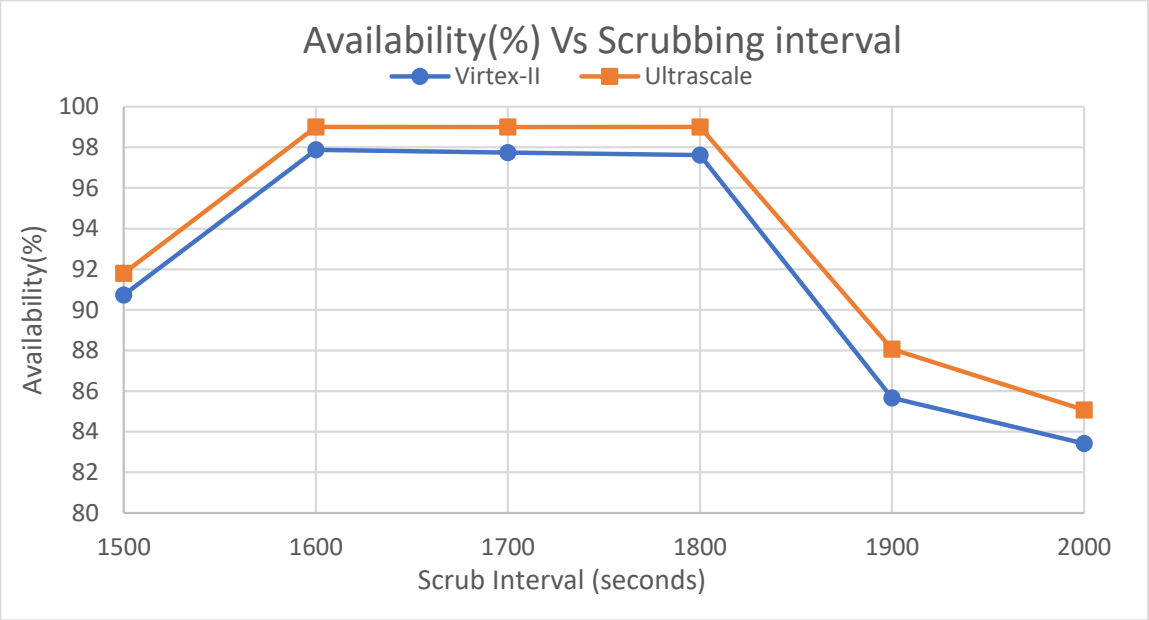

Figure 4.5: Availability(%) Vs scrub interval

The results in Fig.4.5 shows the impact of the scrubbing interval on the system's availability. As the scrubbing interval increase, so does the availability of the components until 1800 seconds. The peak availability is betwen 1600 and 1800 seconds for both devices. Increasing the scrubbing interval beyond this points deteriorates the overall availability.

4.3.3 Experiment 2: UAV-GCS Availability and scrub time

In this experiment, we look at the effect of the time it takes to scrub a device on the overall availability. First, we set the scrubbing interval to 1500 seconds. Also, the scrubbing time is initialized to 1 second. For each run in the experiment, scrub time is increased by 1 second up to 5 seconds.

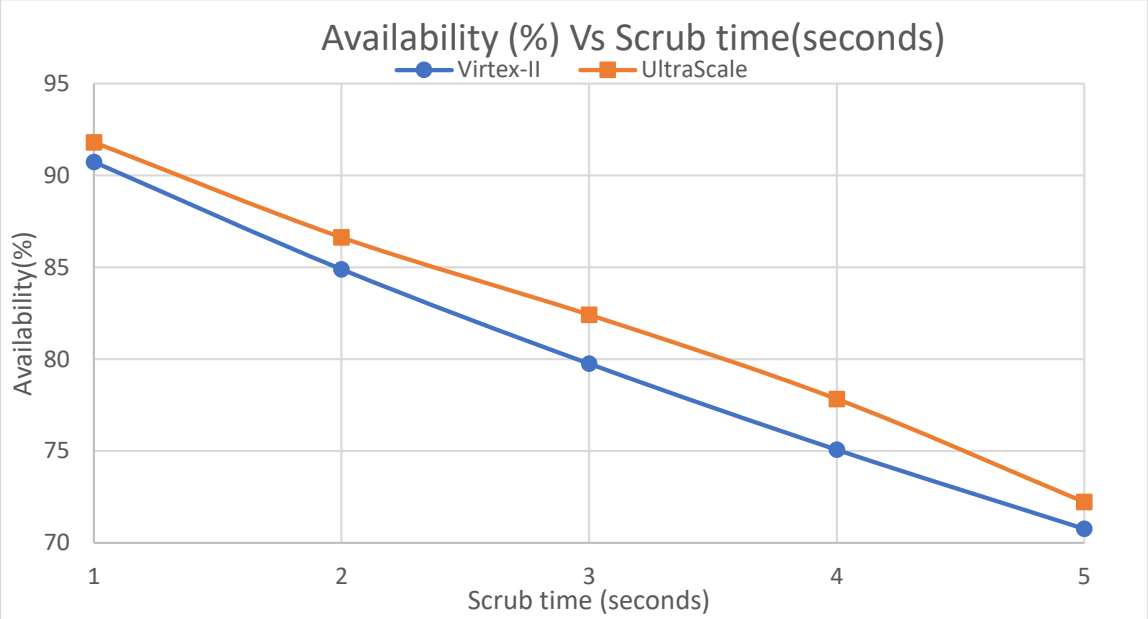

Figure 4.6: Availability(%) Vs scrub interval

For both devices, increasing the scrub time dramatically reduces the availability, as shown in Fig.4.6. Also, the scrub time has a minimum impact on the difference between the availability of Virtex-II and Ultrascale device at low and high. The difference is more noticeable when the FPGA requires 3 and 4 seconds to scrub because the λ_{device} now plays an essential role in determining the availability along with the scrubbing time. The maximum difference in the availability between Virtex-II and UltraScale is 2.76% when the FPGA requires 4 seconds to reconfigure itself.

4.3.4 Experiment 3: UAV-GCS Availability and Number of Missions

This experiment aims to measure the effect of changing the number of communicated messages and missions on the availability of UAV-GCS communication using the Virtex-II FPGA device. The scrubbing interval and scrub time are set to 1500 and 3 seconds, respectively. The number of messages for each mission starts from 10 messages and is incremented by 10 messages up to 50 messages. This experiment was conducted for 5 missions giving us a total of 250 communicated messages at 50 messages. We update the number of missions from 5 missions to 10 messages to evaluate the effect of the number of mission on the availability.

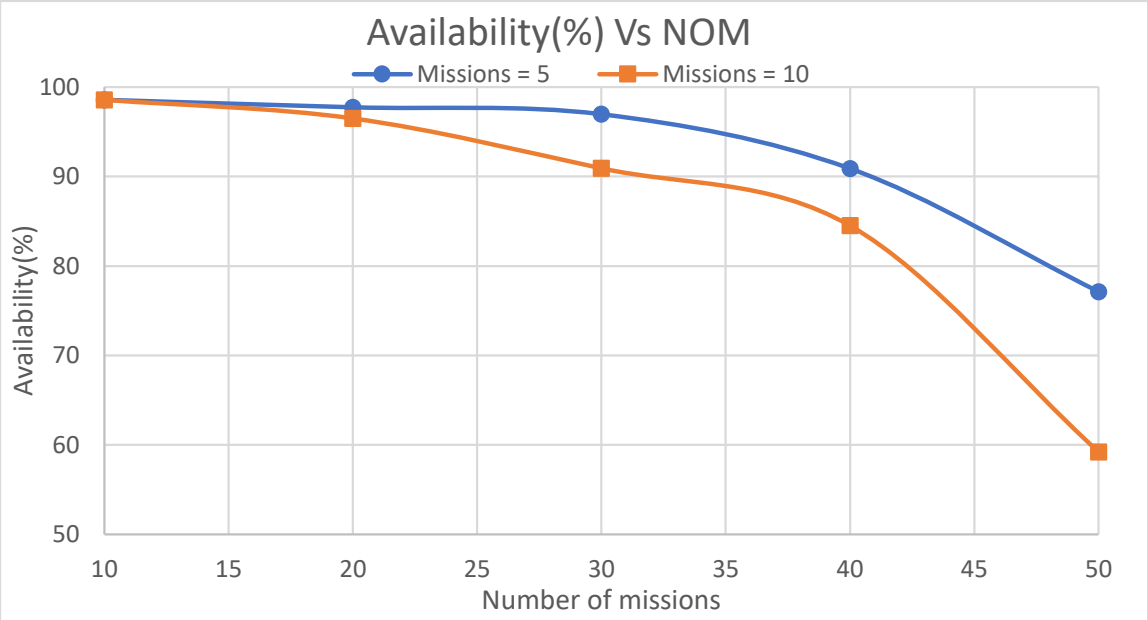

Figure 4.7: Availability(%) Vs Number of Missions.

Two crucial conclusions can be drawn from Fig.4.7. First, increasing the number of exchanged items severely impacts the availability of communication. For example, at 5 missions and 10 messages, the availability is almost 99%, whereas, at 50 messages, the availability drops to almost 78%. Also, a low number of messages, the system's availability is similar regardless of the mission count.

4.3.5 Experiment 4: UAV-GCS Non catastrophic failure

Testing the non-catastrophic failure induced by SEU faults on the UAV communication is extremely complex as it can go unnoticed till it has a negative impact on the system. Therefore, to highlight the devastating repercussions of such failures we assume that any non-catastrophic failure modifies the message received by the UAV from the GCS. This change in the message increase the acceleration speed of the UAV. Also, it is assumed that re-configuring the FPGA occurs without affecting the availability of the communication component. In this experiment, the UAV is required to complete 10 missions each of these mission is made of 10 messages. Scrubbing the FPGA requires 2 seconds and the interval between each scrub is fixed to 50 seconds.

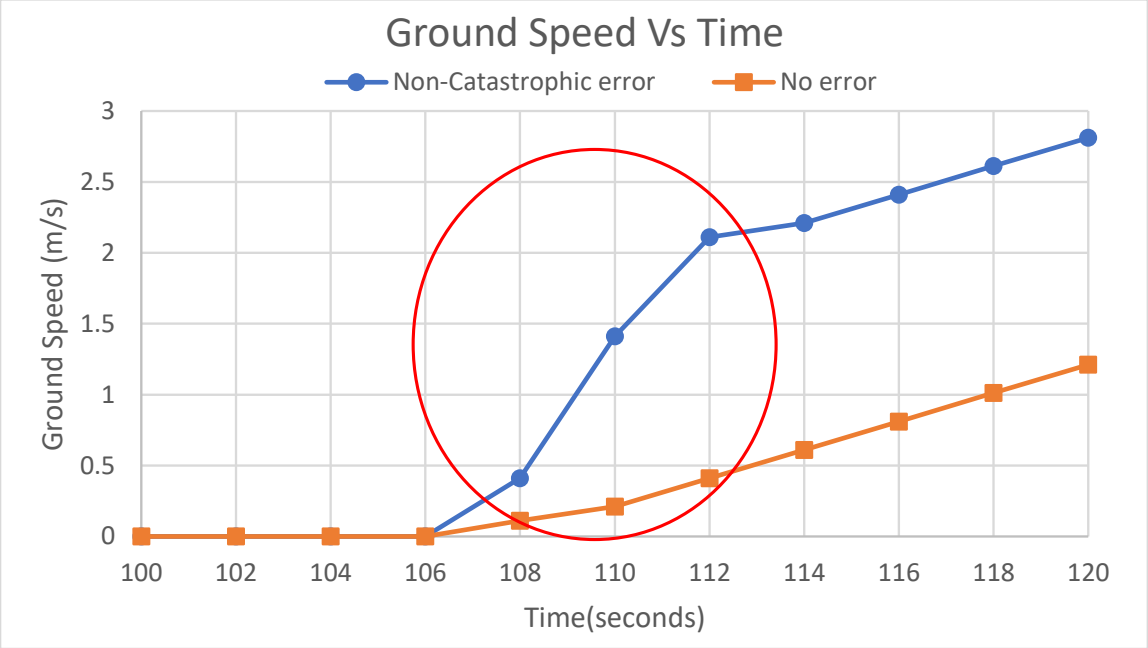

Figure 4.8: Non-catastrophic failure and regular operation.

Comparing the graphs for a normal operating UAV speed and the speed of an UAV with failures in Fig.4.8 shows that there is an enormous difference in the speeds of the two UAVs between 108 and 112 seconds. After 112 seconds the UAV is scrubbed from the non-catastrophic error and therefore, it returns back to normal acceleration. However, this unpredictable substantial burst in the speed for just 4 seconds brings about disastrous issues for the UAV like crashing into other UAVs or even nearby things in the environment.

4.3.6 Experiment 5: UAV-UAV Availability and scrubbing interval

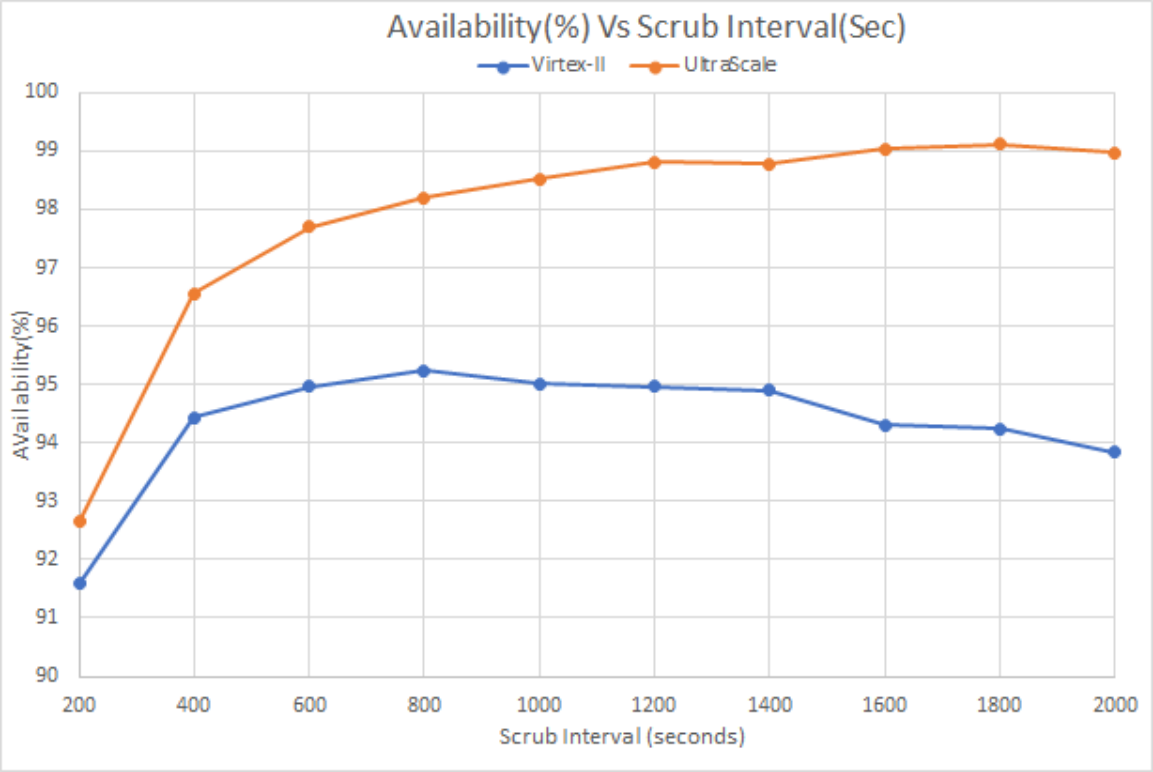

Figure 4.9: UAV-UAV Availability Vs Scrub time.

The effect of the blind scrubbing technique on the UAV-UAV communication availability is measured in this experiment. Any bottleneck present in the communication between UAVs due to received signal strength is limited by setting the antenna gains to 3.2 and P_t of both UAVs to 4 dBm. It is assumed that 10 messages is required to be exchanged after establishing the connection for a a replacement to occur between U1 and U2 successfully. Similar to UAV-GCS experiment we first testing the blind scrubbing effect on Virtex.II. Hence, λ_device is set to to adhere Table.3. At first, we set the scrubbing interval to 200 seconds. Then, for each subsequent run we increase the scrubbing interval by 200 seconds up to 2000 seconds. Each time the scrubbing is triggered the FPGA takes 1 second to reconfigure itself. To compare the effect of changing devices on the availability, The experiment is repeated using the same scrub interval incrementation but for the UltraScale device.

The results plotted in Fig.4.9 show that the availability of the UltraScale and

Virtex-II devices varies based on the chosen scrub interval. Increasing the time between scrub positively influence the overall system availability. The difference in the availability between the devices is minimal when the scrubbing interval is 200 seconds. This is because the communication components are not constantly disturbed by the reconfiguration of the devices. Choosing this range for the scrubbing interval allows us to measure the effect when the availability is 90% or more. The peak availability range is between 800 and 1200 seconds for the Virtex-II. For UltraScale, the peak availability starts from 1200 seconds and continues till 2000 seconds. However, further increasing the scrubbing interval decreases the UAV-UAV communication availability. This can be seen in the Virtex-II device where the availability becomes lower when the scrubbing interval goes beyond 1200 seconds.

4.3.7 Experiment 6: UAV-UAV Availability and scrub time

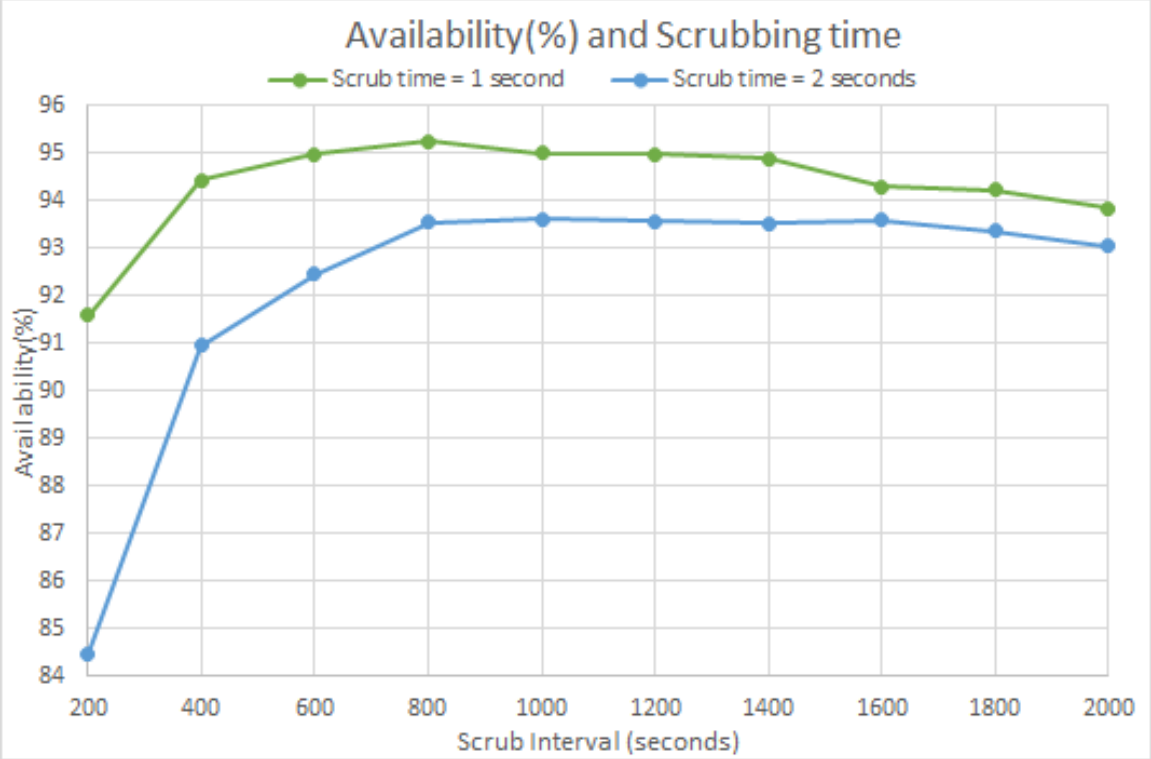

Figure 4.10: UAV-UAV Availability Vs scrub interval.

Using the Virtex-II device, we evaluate the effect of the scrub time on availability. This experiment updates the scrub time to 2 seconds and maintains the incremental

steps of the scrubbing interval shown in the above experiment. From Fig.4.9 it is shown that the availability varies drastically at low scrubbing intervals. For instance, at 200 seconds the difference between 1 second and 2 seconds is approximately 7%. Increasing the scrubbing interval minimizes the impact of the time taken to scrub the device. Mainly this is due to the amount of time where the system is unavailable due to scrubbing. Hence, the number of scrubs can be decreased by increasing the time between two scrubs.

4.4 Comparative Experiment

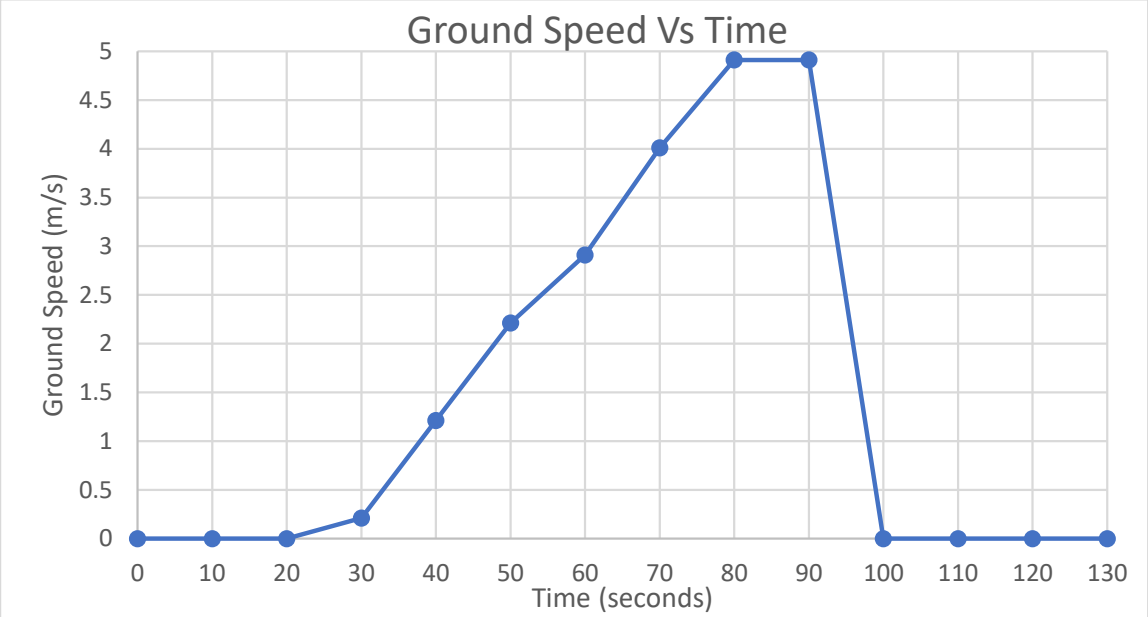

Figure 4.11: Ground speed direct SEU fault injection.

The main objective of this experiment is to examine the viability of our framework. Evaluating the viability is achieved by comparing our work with the results shown in [29]. This paper empirically analyzed the vulnerability of different types of attacks, including packet-injection attacks. They were inserting a packet injection attack when the UAV and GCS are communicating. This attack leads to a halt in the mission by forcing the UAV to hover in its current position. A package injection attack is a malignant message given to the UAV that changes the message type when the UAV and GCS exchange messages. Conceptually this effect is similar to a if a MBU occurs

in the UAV. For example, a change in a few bits can materialize due to the radiation effect in the header of the MAVlink message, and therefore, the message type can be modified due to soft faults. Hence, these faults are similar enough to be viewed as package injection attacks.

Looking at the graph presented in Fig.4.11, we can see that the UAV receives every message initially communicated before 90 seconds correctly. This is confirmed by the increase of speed up to this point. However after 90 seconds, Gs falls to 0 m/s. The drop in Gs demonstrates that a catastrophic fault is successfully inserted, and the UAV is currently hovering in the current coordinates.

We observe that the overall trend and results fall in line with the behavior of the packet injection experiment from [29]. Our methodology provides several advantages when performing these experiments. First, our approach allows for several protocols to be tested without any physical components making it very flexible and cost-efficient. Additionally, a considerable number of experiments can be performed every day. Therefore, our technique produces results using a fraction of the needed cost and time.

4.5 Discussion

We discuss key elements that severely impact the reliability or availability of an UAV communication protocol based on the results obtained from sections 4.3 and 4.4. We first discuss the effect of the communication parameters on the reliability of the UAV-UAV link. Then, we elaborate on the impact of changing various variables on the availability of the UAV-UAV and UAV-GCS communication. Finally, we discuss the limitations of our methodology based on these observations.

4.5.1 UAV-UAV framework discussion

P_t and antenna gains

Based on the results presented in Fig.4.2, the effectiveness of G_t and G_r become less prevalent at higher P_t values. Testing the system at 4 dBm shows that the reliability is similar despite the difference in antenna gain used for each experiment. Another interesting remark extracted from this figure is that increasing P_t does not

always benefit the link's reliability. Such behavior can be noticed when P_t jumps from 2.0 to 2.4 dBm. Also, when the gain is 3.2, the failure rate remains the same (7%) at 3.2 dBm and 3.6 dBm. Therefore, a designer can improve the battery life of an UAV by using communication components with high antenna gains at low P_t levels. Additionally, if the mission is crucial, the battery life can be sacrificed slightly to achieve maximum reliability in the system.

Number of messages and transmission delay

If a communication component has high transmission delay, the UAV can only send or receive a small number of messages before the link cuts-off. Therefore, only the crucial information must be sent in this case. Changing the device use in data transmission at lower delay, can increase the number of data transferred if necessary. This overall behavior is proven in Fig.4.3. This however, raises the cost of the UAV.

Node technology effect on the availability

The node technology severely impacts the overall system availability as shown in Fig.4.10. Several conclusions can be drawn from this experiment. First, the scrubbing interval peak for with lower has better overall availability. Second, the scrubbing interval peak range shifts as λ_device. For instance, the peak scrubbing interval for Virtex-II is between 800-1200 seconds whereas UltraSCale peaks at 1200 seconds and continues up to 2000 seconds. Hence, the range shifted from 800 to 1200 seconds and the range is extended by 200 seconds for the UltraScale peak availability. Finally, decreasing scrubbing interval duration leads to an influx in the number of scrubs and therefore, interrupts the communication between the UAV and GCS.

4.5.2 UAV-GCS framework discussion

Blind scrubbing and availability

SEUs is assumed to only affect the UAV. Therefore, when it is no transmitting or receiving a message, the fault of the SEU is masked. Additionally, if scrubbing occurs when the UAV is idle, the system can reconfigure without dropping any message. Hence, the overall system availability increases.

Number of missions, messages and availability

From Fig.4.7 we notice that increasing the number of missions have minimal impact on the system for a small number of messages. Whereas, increasing the number of messages drastically reduces the availability. Therefore, it is best to have a large number of missions, each mission contains few messages to increase the availability.

Effect of blind scrubbing on Non Catastrophic failure

The distance between UAVs is very critical especially when a huge number of them is deployed for a single mission. Therefore, extreme non-catastrophic failures like the one presented in Fig.4.8 is detrimental in terms of mission correctness and safety. Based on this experiment, it is clear that the blind scrubbing technique will hinder the availability of the system if it is used to deal with such errors. This is because of the short duration between scrubs that is necessary to catch these faults.

4.5.3 Limitations

As our work is still a high level abstraction of different physical components of a system in our frameworks. Therefore, the limitations of our methodology can be divided into three main categories:

- **General limitations:**

  Broadly speaking our methodology suffers from two main limitations for both the UAV-UAV and UAV-GCS framework. First, converting a communication protocol to a UPPAAL-SMC compatible format is tremendously time consuming. This is because ant protocol needs to be first abstracted and converted to a finite state machine. Also, the PTA needs to be thoroughly checked to avoid any unnecessary abstractions. Additionally, the SEU rate presented in our work is only an estimation of the worst-case SEU rate that can be expanded upon further.

- **UAV-UAV framework limitations:**

  The UAV-UAV communication framework experiences two primary limitations. To begin with, the UAV speed calculated in this framework is a high level abstraction for a lot of variables such as wind speed, air drag and other weather

conditions. Moreover, this framework focuses on the point-point communication of the UAVs in a swarm. However, it lacks the ability to test for UAV-many communication protocols.

- **UAV-GCS framework limitations:**

  Currently three limitations are present in the UAV-GCS framework. First, we have the scrubbing techniques. While the blind scrubbing technique is a classical approach that can solve many SEU related faults quickly and reliably it can lag behind other scrubbing techniques in terms of terms of power consumption and area. Also, blind scrubbing is not a viable approach to solve non-catastrophic faults. Moreover, we assume that any mission item received by the UAV is for motion in the x-axis. However, the types of mission exchanged can be expanded upon in future work. Finally, it is assumed that the GCS is fully protected from any SEU fault that can occur.

4.6 Summary

In this chapter, we presented four experiments to measure the reliability of the replacement negotiation scenario using the UAV-UAV framework. These experiments indicate that P_t and the antenna gains have a dominant effect on the reliability. Additionally, the data exchange phase is susceptible to transmission delay as the number of exchanged messages between U1 and U2 increases. After that, we calculate the first-order worst-case SEU rate for the Xilinx Virtex-II and UltraScale products. These devices were used to perform 4 additional experiments to measure the availability of the MAVlink communication flow through the UAV-GCS framework. Moreover, the availability of the replacement negotiation scenario is studied using the UAV-UAV framework. Then, we performed a viability experiment by comparing the UAV-GCS framework with the available results in the literature. After that, these results were used to deliberate on the relation between the different parameters on the system's dependability. Finally, we elaborated on the limitations of our work based on performed experiments and discussion.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

This thesis proposed a methodology to evaluate the reliability and availability of a FPGA-based UAV communication protocol using statistical model checking through the UPPAAL-SMC tool. In chapter 3, we first discuss our overall methodology. Then, we showcase the UAV-UAV framework to measure the reliability of a UAV-UAV communication protocol. This framework is composed of several PTAs that work together to emulate the the motion of a UAV in the air, the condition of the communication device (transmitter and receiver) in the presence of SEU, and the data exchange phase between two UAVs. After that, we improved our UAV-UAV framework by removing the abstractions in the PTAs of the transmitter and receiver regarding the the blind scrubbing technique. To do that, we first implemented the RTL code for the serial communication component to calculate the first-order worst-case scenario SEU rate of a communication module. Then, we re-designed the transmitter and receiver PTAs to implement the scrubbing interval and scrub time.

Our overall methodology was further expanded upon by designing and building another framework to assess the effect of blind scrubbing on the availability of a UAV-GCS link. In this approach, we introduce several types of failures that can manifest due to SEUs. Finally, we introduced and discussed the replacement negotiation scenario and MAVlink communication flow PTAs as case study protocols for our frameworks.

The reliability and availability of the case study communication protocols were

analyzed through our methodology in Chapter 4. First, we evaluated the reliability of the replacement negotiation scenario using Bluetooth specifications as a case study telemetry device. The results from the experiments indicate that at low transmission power, the antenna gain is crucial to maintain acceptable link reliability. Additionally, P_t must be 3.2 dBm or higher to maintain 90% or more reliability at any tested antenna gain. Then, we calculated the SEU rate of the Virtex-II and UltraScale as case study implementation FPGAs. Utilizing this SEU rate, we measured the availability of the MAVlink communication flow and replacement negotiation scenario using blind scrubbing as an SEU mitigation technique for an FPGA-based UAV. From the experiments, we learned that for a UAV-UAV communication component, the availability rises as the scrubbing interval increase. Also, communication components that has low SEU rates can maintain the peak availability for longer periods of times. Finally, we examined the effect of blind scrubbing on an UAV that sends and receives commands from a GCS. The results obtained from these experiments show that the optimum scrubbing interval for a UAV-GCS communication module is when the UAV communication module is idly waiting for a message to be received from the GCS.

5.2 Future Work

FPGA-based UAVs still have a long way ahead of them before they can be commercially realized. Similarly, the verification of the components involved in these devices is still in its infancy and can be evolved and improved. Our methodology presents a foundation for future improvements in the area of analysis for UAV communication components and their protocols. However, based on the discussion presented in Section 4.4 of Chapter 4, several improvements can be made to add more features and make it more robust. The list below highlights some of the future works that can further improve the framework:

- Creating a tool to convert a FSM into XML compatible file. This can then be immediately read into UPPAAL-SMC. This will help simplify the process of converting a protocol to a PTA.
- Fault injection campaigns can be initiated to obtain a more accurate representation of the SEU rate for a device.

- Implementing a broadcast PTA to further expand the UAV-UAV framework. Using this PTA the framework can analyze UAV-many communication protocols like the MUSCOP protocol. Additionally, a Speed PTA can be built to realize a more accurate representation for the UAV motion in air for the UAV-UAV framework.
- Apply other scrubbing techniques as PTAs in the UAV-GCS framework to measure their effect on the availability and non-catastrophic failure due to SEUs.
- Build a simulation environment in AEROSTACK and use it as a benchmark for the frameworks.

Bibliography

[1] Ayman Ahmed. New fpga blind scrubbing technique. In *2016 IEEE Aerospace Conference*, pages 1–9. IEEE, 2016.

[2] Marwan Ammar, Ghaith Bany Hamad, Otmane Ait Mohamed, and Yvon Savaria. System-level analysis of the vulnerability of processors exposed to single-event upsets via probabilistic model checking. *IEEE Transactions on Nuclear Science*, 64(9):2523–2530, 2017.

[3] Hossein Asadi, Mehdi B Tahoori, Brian Mullins, David Kaeli, and Kevin Granlund. Soft error susceptibility analysis of sram-based fpgas in high-performance information systems. *IEEE Transactions on Nuclear Science*, 54(6):2714–2726, 2007.

[4] Algirdas Avizienis, Jean-Claude Laprie, and Brian Randell. Fundamental concepts of dependability. *Department of Computing Science Technical Report Series*, 2001.

[5] Ran Bao, Christian Attiogbe, Benoit Delahaye, Paulin Fournier, and Didier Lime. Parametric statistical model checking of uav flight plan. In *International Conference on Formal Techniques for Distributed Objects, Components, and Systems*, pages 57–74. Springer, 2019.

[6] Mona Batra. Formal methods: Benefits, challenges and future direction. *Journal of Global Research in Computer Science*, 4(5):21–25, 2013.

[7] Ilker Bekmezci, Ozgur Koray Sahingoz, and Şamil Temel. Flying ad-hoc networks (fanets): A survey. *Ad Hoc Networks*, 11(3):1254–1270, 2013.

[8] Peter Bulychev, Alexandre David, Kim Gulstrand Larsen, Marius Mikučionis, Danny Bøgsted Poulsen, Axel Legay, and Zheng Wang. Uppaal-smc: Statistical

model checking for priced timed automata. *arXiv preprint arXiv:1207.1272*, 2012.

[9] Edmund M Clarke. The birth of model checking. In *25 Years of Model Checking*, pages 1–26. Springer, 2008.

[10] Edmund M Clarke Jr, Orna Grumberg, Daniel Kroening, Doron Peled, and Helmut Veith. *Model checking*. MIT press, 2018.

[11] Alexandre David, Kim G Larsen, Axel Legay, Marius Mikučionis, and Danny Bøgsted Poulsen. Uppaal smc tutorial. *International Journal on Software Tools for Technology Transfer*, 17(4):397–415, 2015.

[12] Blanca de Miguel Molina and Marival Segarra Oña. The drone sector in europe. *Ethics and civil drones*, pages 7–33, 2018.

[13] Balbir S Dhillon. *Reliability in computer system design*. Intellect Books, 1987.

[14] Thomas Dietrich, Oleksandr Andryeyev, Armin Zimmermann, and Andreas Mitschele-Thiel. Towards a unified decentralized swarm management and maintenance coordination based on mavlink. In *2016 International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 124–129. IEEE, 2016.

[15] Paul E Dodd and Lloyd W Massengill. Basic mechanisms and modeling of single-event upset in digital microelectronics. *IEEE Transactions on nuclear Science*, 50(3):583–602, 2003.

[16] Sophie Duzellier. Radiation effects on electronic devices in space. *Aerospace science and technology*, 9(1):93–99, 2005.

[17] Mario Arturo Ruiz Estrada and Abrahim Ndoma. The uses of unmanned aerial vehicles–uav's-(or drones) in social logistic: Natural disasters response and humanitarian relief aid. *Procedia Computer Science*, 149:375–383, 2019.

[18] Francisco Fabra, Willian Zamora, Pablo Reyes, Julio A Sanguesa, Carlos T Calafate, Juan-Carlos Cano, and Pietro Manzoni. Muscop: Mission-based uav swarm coordination protocol. *IEEE Access*, 8:72498–72511, 2020.

[19] Blake Fuller, Jonathan Kok, Neil Kelson, and Felipe Gonzalez. Hardware design and implementation of a mavlink interface for an fpga-based autonomous uav flight control system. In *Proceedings of the 16th Australasian Conference on Robotics and Automation 2014*, pages 1–6. Australian Robotics and Automation Association Inc., 2014.

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

[21] K. A. Hoque et al. Probabilistic model checking based dal analysis to optimize a combined tmr-blind-scrubbing mitigation technique for fpga-based aerospace applications. In *2014 Twelfth ACM/IEEE Conference on Formal Methods and Models for Codesign (MEMOCODE)*, pages 175–184, 2014.

[22] Khaza Anuarul Hoque, O Ait Mohamed, Yvon Savaria, and Claude Thibeault. Probabilistic model checking based dal analysis to optimize a combined tmr-blind-scrubbing mitigation technique for fpga-based aerospace applications. In *2014 Twelfth ACM/IEEE Conference on Formal Methods and Models for Codesign (MEMOCODE)*, pages 175–184. IEEE, 2014.

[23] Yu-Shun Hsiao, Zishen Wan, Tianyu Jia, Radhika Ghosal, Arijit Raychowdhury, David Brooks, Gu-Yeon Wei, and Vijay Janapa Reddi. Mavfi: An end-to-end fault analysis framework with anomaly detection and recovery for micro aerial vehicles. *arXiv preprint arXiv:2105.12882*, 2021.

[24] Ricki G Ingalls. Introduction to simulation. In *Proceedings of the 2011 Winter Simulation Conference (WSC)*, pages 1374–1388. IEEE, 2011.

[25] A H M Jakaria and Mohammad Ashiqur Rahman. Formal analysis of k-resiliency for collaborative uavs. In *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, volume 01, pages 583–592, 2018.

[26] Ralf Joost and Ralf Salomon. Advantages of fpga-based multiprocessor systems in industrial applications. In *31st Annual Conference of IEEE Industrial Electronics Society, 2005. IECON 2005.*, pages 6–pp. IEEE, 2005.

[27] P. Kenterlis, N. Kranitis, A. Paschalis, D. Gizopoulos, and M. Psarakis. A low-cost seu fault emulation platform for sram-based fpgas. In *12th IEEE International On-Line Testing Symposium (IOLTS'06)*, pages 7 pp.–, 2006.

[28] Anis Koubâa, Azza Allouch, Maram Alajlan, Yasir Javed, Abdelfettah Belghith, and Mohamed Khalgui. Micro air vehicle link (mavlink) in a nutshell: A survey. *IEEE Access*, 7:87658–87680, 2019.

[29] Young-Min Kwon, Jaemin Yu, Byeong-Moon Cho, Yongsoon Eun, and Kyung-Joon Park. Empirical analysis of mavlink protocol vulnerability for attacking unmanned aerial vehicles. *IEEE Access*, 6:43203–43212, 2018.

[30] Kenneth A Label et al. Single event effect criticality analysis. *NASA Headquarters/Code QW*, 1996.

[31] D. S. Lee et al. Single-event characterization of the 20 nm xilinx kintex ultrascale field-programmable gate array under heavy ion irradiation. In *2015 IEEE Radiation Effects Data Workshop (REDW)*, pages 1–6, 2015.

[32] Axel Legay, Benoît Delahaye, and Saddek Bensalem. Statistical model checking: An overview. In *International conference on runtime verification*, pages 122–135. Springer, 2010.

[33] Peter Lindenfeld and Suzanne White Brahmia. *Physics: the first science*. Rutgers University Press, 2011.

[34] Ian A Mason, Vivek Nigam, Carolyn Talcott, and Alisson Brito. A framework for analyzing adaptive autonomous aerial vehicles. In *International Conference on Software Engineering and Formal Methods*, pages 406–422. Springer, 2017.

[35] Steve Mills. *The Dawn of the Drone: From the Back-Room Boys of World War One*. Casemate, 2019.

[36] UM Rao Mogili and BBVL Deepak. Review on application of drone systems in precision agriculture. *Procedia computer science*, 133:502–509, 2018.

[37] Mahsa Mousavi, Hamid Reza Pourshaghaghi, Mohammad Tahghighi, Roel Jordans, and Henk Corporaal. A generic methodology to compute design sensitivity

to seu in sram-based fpga. In *2018 21st Euromicro Conference on Digital System Design (DSD)*, pages 221–228. IEEE, 2018.

[38] S. Nethula. Single event multiple transient analysis using satisfiability modulo theories. Master's thesis, Concordia University, 2019.

[39] Nicholas Neveu, Yang-Ki Hong, Jaejin Lee, Jihoon Park, Gavin Abo, Woncheol Lee, and David Gillespie. Miniature hexaferrite axial-mode helical antenna for unmanned aerial vehicle applications. *IEEE transactions on magnetics*, 49(7):4265–4268, 2013.

[40] Barrett O'Neill. Chapter 2 - frame fields. In Barrett O'Neill, editor, *Elementary Differential Geometry (Second Edition)*, pages 43 – 99. Academic Press, Boston, second edition edition, 2006.

[41] Enrico Petritoli, Fabio Leccese, and Lorenzo Ciani. Reliability assessment of uav systems. In *2017 IEEE International Workshop on Metrology for AeroSpace (MetroAeroSpace)*, pages 266–270. IEEE, 2017.

[42] John C Porcello. Designing and implementing ofdm communications for advanced multifunction uav payloads using fpgas. In *2012 IEEE Aerospace Conference*, pages 1–12. IEEE, 2012.

[43] Theodore S. Rappaport. *Wireless Communications Principles and Practice*. Pearson, 2 edition, 2001.

[44] ArduPilot Dev Team. *Auto Mode*, (accessed December 9th, 2020).

[45] NASA. *Relative Velocities*, (accessed September 29th, 2020).

[46] Falk Salewski and Adam Taylor. Fault handling in fpgas and microcontrollers in safety-critical embedded applications: A comparative survey. In *10th Euromicro Conference on Digital System Design Architectures, Methods and Tools (DSD 2007)*, pages 124–131. IEEE, 2007.

[47] Jose Luis Sanchez-Lopez, Ramón A Suárez Fernández, Hriday Bavle, Carlos Sampedro, Martin Molina, Jesus Pestana, and Pascual Campoy. Aerostack: An architecture and open-source software framework for aerial robotics. In *2016*

International Conference on Unmanned Aircraft Systems (ICUAS), pages 332–341. IEEE, 2016.

[48] Felix Schill, Alexander Bahr, and Alcherio Martinoli. Vertex: A new distributed underwater robotic platform for environmental monitoring. In *Distributed Autonomous Robotic Systems*, pages 679–693. Springer, 2018.

[49] Reza Shakeri, Mohammed Ali Al-Garadi, Ahmed Badawy, Amr Mohamed, Tamer Khattab, Abdulla Khalid Al-Ali, Khaled A Harras, and Mohsen Guizani. Design challenges of multi-uav systems in cyber-physical applications: A comprehensive survey and future directions. *IEEE Communications Surveys & Tutorials*, 21(4):3340–3385, 2019.

[50] J.F. Wakerly. Microcomputer reliability improvement using triple-modular redundancy. *Proceedings of the IEEE*, 64(6):889–895, 1976.

[51] Wonsang Yoo, Eun Yu, and Jaemin Jung. Drone delivery: Factors affecting the public's attitude and intention to adopt. *Telematics and Informatics*, 35(6):1687–1700, 2018.

[52] Zhenhui Yuan, Xiwei Huang, Lingling Sun, and Jie Jin. Software defined mobile sensor network for micro uav swarm. In *2016 IEEE international conference on control and robotics engineering (ICCRE)*, pages 1–4. IEEE, 2016.

Biography

Education

- **Concordia University**: Montreal, Quebec, Canada.
  MASc., Electrical & Computer Engineering (September 2019 - May 2021)
- **United Arab Emirates University**: Alain, United Arab Emirates.
  BAsc., Electrical Engineering (September 2014 - December 2019)

Awards

- Concordia University, Concordia Merit Scholarship (MASc. Program).
- United Arab Emirates , Four Academic Awards.

Work History

- **Research Assistant**, Hardware Verification Group, Department of Electrical and Computer Engineering, Concordia University, Montreal, Quebec, Canada (2019-2021).

Publications

Conference Papers

- **Bio-Cf1** M.Abdelhamid, A.Attallah, M.Ammar, and O.Ait Mohamed, "Reliability Analysis Of Autonomous UAV communication using statistical model checking," Accepted at 2021 IEEE 64th International Midwest Symposium on Circuits and Systems..

- **Bio-Cf2** M.Abdelhamid, A.Attallah, M.Ammar, and O.Ait Mohamed, "PTA based availability analysis of the effects ofblind scrubbing of UAV-UAV communication usingSRAM based FPGAs," Accepted at RADECS conference 2021.

- **Bio-Cf3** Abdrabou, A., Darwish, M., Dalao, A., AlKaabi, and M.Abutaqiya M. (2020). Energy-Aware WiFi Network Selection via Forecasting Energy Consumption. International Journal of Electronics and Telecommunications, 66(2), 339-345.

- **Bio-Cf4** A. Abdrabou, M. Prakash, A. S. AlShehi, S. Ahmed and M. Darwish, "An Experimental Study on Energy Consumption of Wireless Multipath TCP Connections," 2019 Wireless Telecommunications Symposium (WTS), 2019, pp. 1-7, doi: 10.1109/WTS.2019.8715523.