Experimental Results Analyzes in Resilient Mechanism for SDN-Based UAV Network Applied to Environmental Protection Area Surveillance

Diego S. Pereira[12], **Vitor G. Santos** [1], **Luís B. P. Nascimento**[12], **Pablo J. Alsina**[2]

[1]Federal Institute of Education, Science and Technology of Rio Grande do Norte (IFRN)
Parnamirim – RN – Brazil

[2]Federal University of Rio Grande do Norte (UFRN)
Natal – RN – Brazil

{diego.pereira, vitor.gaboardi}@ifrn.edu.br
luisbrunu@gmail.com, pablo@dca.ufrn.br

Abstract. *A surveillance system requires repetitive and uninterrupted actions, typically related to large extension places, and difficult access. In this context, a multiple Unmanned Aerial Vehicles (multi-UAV) system is a good alternative for overcoming the requirements imposed by this application. With this in mind, the UAVs have to work cooperatively and exchange information to finish the mission. However, managing and keeping the communication between UAVs is a challenge that has been investigated. So, the SD-FANET is an SDN architecture developed to mitigate this communication problem. SD-FANET has a hierarchical distributed control plane that provides a resilience mechanism to overcome failures during a mission runtime. The three-step strategy (detection, election, and recovery) allows the control plane outperforms failures and works continuously as long as there are nodes in the UAV network. Experimental tests were performed in three scenarios. In all of them, the controller executed the resilience mechanism and keep going working. The mean time was 1,94 seconds to 300 executions. The PDF of results was similar to a normal distribution demonstrating the behavior of the recovery time.*

1. Introduction

Surveillance systems are related to monitoring people, behaviors, activities, area access, etc. Among some common methods used for surveillance are face-to-face surveillance, cameras surveillance, GPS tracking surveillance, radio surveillance, satellite surveillance, and biometric surveillance. Tasks are usually related to environmental monitoring, area patrol, construction management, power grid inspection, traffic monitoring, etc [Yue et al. 2018].

A typical environment surveillance system requires repetitive and continuous tasks to reach a satisfactory result. Moreover, manpower and costs are limiting factors to keep the system online, mainly in cases where a wide area must be covered by the surveillance system. An increasingly cheaper, stable, and capable of transporting powerful embedded systems alternative to overcome these issues is Unmanned Aerial Vehicles (UAVs) [Li and Savkin 2021].

Surveillance systems have to be resilient. In this way, systems with only one UAV are usually inappropriate because the mission can be compromised if the UAV fails. In this perspective, a fleet of UAV introduces robustness and resilience to the system since they can collect more data and be attached with different sensors and capabilities, solving more complex missions [Zhou et al. 2020].

In this context, applications involving multiple UAVs, or just multi-UAV, are planned to use a fleet of UAVs capable of acting cooperatively to achieve a common objective [Fu et al. 2019]. Actions to explore unknown areas [Tang et al. 2019], provision of coverage services for telephone networks [Liu et al. 2019], search and rescue in difficult-to-access regions [Alotaibi et al. 2019], environmental monitoring [Vazquez-Carmona et al. 2019], and data collection [Binol et al. 2019] are some examples of applications.

Given the need to exchange information between the aircraft that compose the multi-UAV system, it is necessary to assign the capacity to create and organize an Flying Ad hoc Network (FANET) [Bekmezci et al. 2015]. FANET allows flexibility for the multi-UAV system operation and does not require a prior communication infrastructure, which is important given the dynamic behavior of the aircraft and the intermittence of wireless communication links.

Our previous work [Santos et al. 2019] presented a multi-UAV architecture for monitoring human activity in a Coral Reefs Environmental Protection Area, located in the state of Rio Grande do Norte/Brazil. The main contributions were establishing an UAV network to send images, a flight formation strategy to capture images by UAVs, and evaluating a SSD Neural Network to detect boats in forbidden regions. Figure 1 shows an example of detected boats that should be sent by the UAV network to the base station.

Figure 1. Boat detection in Coral Reefs Environmental Protection Area (Área de Proteção Ambiental de Recifes de Corais - APARC) located in the coastal strip from the cities of Maxaranguape, Rio do Fogo and Touros, State of Rio Grande do Norte/Brazil. Image provide by IDEMA/RN.

However, the hardware components from the proposed architecture have limited requirements to manage the system operation, such as routing configuration restrictions, not allowing traffic priority or any quality of service policies, and lacking failure detection and recovery. These features are essential to offer the possibility of better management.

In this way, we proposed the SD-FANET [Pereira et al. 2019] an SDN-based architecture to overcome these limitations and improve the UAV network performance. According to [Foerster et al. 2018] Software-Defined Networking (SDN) decouples the control plane and data plane, allowing a logically centralized controller to program the network, making the other switching devices that carry out data forwarding from a flow table maintained by the controller.

Thus, the objective of this work is to propose a resilience mechanism for the control plane of the SD-FANET architecture. Besides, we will validate our strategy using environmental protection area surveillance as a case study, where it is essential to secure a continued operation of the communication network and an efficient delivery of data to the ground base station. Therefore, SDN is a good alternative since it supplies flexibility and scalability.

The main contributions of this paper are the development of a resilient strategy for a hierarchical distributed SDN controller and an election procedure to select a new coordinator based on distributed system concepts, and the evaluation of experimental results using a fleet of UAVs with an embedded system to manage the UAV network.

The remainder of this paper is organized as follows: Section II presents the related works, Section III addresses an overview of SD-FANET and details our proposed resilient strategy. Section IV presents the methodology used to validate our solution. Section V presents some experimental results and Section VI presents conclusions and perspectives about future works.

2. Related Works

The development of resilience strategies for UAV networks is still a new topic of research. We will initially evaluate SDN solutions for UAV Networks, and then present some resiliency solutions for SDN controllers.

2.1. Software-Defined UAV Networks

The use of software-defined networking in UAV networks was initially proposed by [Gupta et al. 2015]. For the authors, SDN allows greater flexibility to deal with the dynamic characteristics of this type of networks, such as aircraft movement and intermittent connectivity links, and the limitation of embedded resources, such as batteries. Therefore, SDN eases the deployment and management of applications and services through the ability to program the network.

In this context, we will first present some works concerning research related to SDN deployment in UAV networks. Next, we will compare them with our SD-FANET proposal, which was introduced in [Pereira et al. 2019]. This comparison is summarized in Table 1. Our proposal is the first that presents a resilient strategy with experimental results.

Table 1. Comparison between UAV network SDN-based architectures.

Architecture	Controller Design	Resilient Controller	Experimental Results
SD-UAVNet [Zhao et al. 2019]	Centralized	No	No
VOEI [Cumino et al. 2018]	Centralized	No	No
SDN-Based FASNET [Qi et al. 2017]	Distributed	No	No
STFANET [e Silva et al. 2019]	Centralized	No	No
SUV [Hu et al. 2021]	Distributed	Yes	No
SD-FANET	Distributed	Yes	Yes

The SD-UAVNet [Zhao et al. 2019] architecture uses the network management from a centralized controller. The paper proposes optimal positions for relay UAVs to keep surveillance services operational. The controller considers the context information from a global UAV to optimize the movements of the other UAVs, such as appropriately selecting routes for the given flows and preventing collisions between aircraft. The goal is to ensure satisfactory video quality for the service.

Similarly, the VOEI [Cumino et al. 2018] architecture features a cooperative scheme between UAVs to improve video transmission and overall system energy efficiency. The main goal is to keep streaming with QoE (Quality of Experience) support. Decisions are based on a centralized controller node that selects reliable routes, increases energy efficiency, and detects the appropriate time to replace a UAV, ensuring a higher system autonomy.

The authors of the STFANET [e Silva et al. 2019] architecture use algorithms to establish and maintain a FANET that provides a constant and reliable communication link between independent nodes that are performing individual or collaborative missions. FANET is composed of relays UAVs that are positioned by a centralized controller, thus managing the link's topology and coverage area. For validation purposes, a simulation was made for military applications.

The SDN-based FASNET [Qi et al. 2017] architecture was designed based on clustering controllers from hierarchical management. The architecture seeks to secure specific QoS (Quality of Service) requirements, sensitive to delays and reliability. Thus, weights are assigned to the flows according to their sensitivity to delay and priority level. Unlike previous proposals, FASNET has a distributed control plane.

The SUV [Hu et al. 2021] architecture proposes the use of blockchain to enable a logically centralized control plane but physically distributed, which is responsible for enabling routing and management of the UAV network. SUV has resilience and safety, features not found in the previous architectures discussed. The paper presents the architecture and its components, however, it does not present any results.

The SD-FANET architecture aims to maintain connectivity between UAV network nodes in surveillance missions. SD-FANET adopts distributed hierarchical controller to optimize network performance and a control plane with failure recovery capability. In this process, a relay UAV is promoted to a controller through an election process among all the UAVs which compose the multi-UAV system. In the next section, we will

present the SD-FANET architecture and detail the resilience mechanisms for failures in the distributed control plane.

2.2. Resilient SDN Controller

As previously discussed, SDN offers conditions to overcome some challenges imposed on UAV networks. However, it is important to highlight that a failure in the control plane can cause interruption of the entire system. Therefore, it is essential to invest in solutions that mitigate and handle failures in the network controller [Botelho et al. 2014a]. In this way, we present below some solutions for this purpose. The details are presented in Table 2.

Table 2. Distributed resilient controllers solutions details.

Paper	Design	Coordinator	Metric	Results
[Botelho et al. 2014b]	Hierarchical	Leader	-	Simulated
[Lakhani and Kothari 2020]	Flat	Leader	Load	Simulated
[Moazzeni et al. 2019]	Hierarchical	Leader	Reliability	Numerical
[Ammar et al. 2017]	Flat	Leader	Load	Simulated

The authors in [Botelho et al. 2014b] present a hierarchical architecture called SMaRtLight. The solution uses a shared replicated database that stores the entire state of the network. A smooth transition mechanism is implemented in case of control failures and avoids the need for an additional coordination service.

The paper of [Lakhani and Kothari 2020] present the Distributed Controller Fault Tolerance (DCFT) method. It is an election algorithm for control failures in which the main metric is load balancing between network controllers. The proposed method uses a flat model.

The reliability improvement of the software-defined networks method is presented in [Moazzeni et al. 2018]. This strategy splits the network into subnets and assigns a controller to each of them. This assignment is done by a general coordinator of the network. The authors also use election algorithms to select a new controller based on a reliability metric [Moazzeni et al. 2019].

Finally, a dynamic workload distribution for controllers in [Ammar et al. 2017] is proposed. From the election of a root controller responsible for coordinating the entire network, a decision-making strategy for load balancing takes place to improve network performance.

3. RESILIENT CONTROLLER FOR SDN-BASED UAV NETWORK

3.1. SD-FANET Overview

The main feature of the SD-FANET architecture is the implementation of a distributed hierarchical controller. The control plane is composed of two components: the BS-Controller (Base Station - BS) and the UAV-Controller. The data plane is composed of UAVs-Relay. However, the UAV-Controller can also forward data. An illustration of the SD-FANET architecture is shown in Figure 2.

The BS-Controller manages the control plane, which imposes the need for a global view of the network and actions to coordinate the other nodes in the distributed controller.

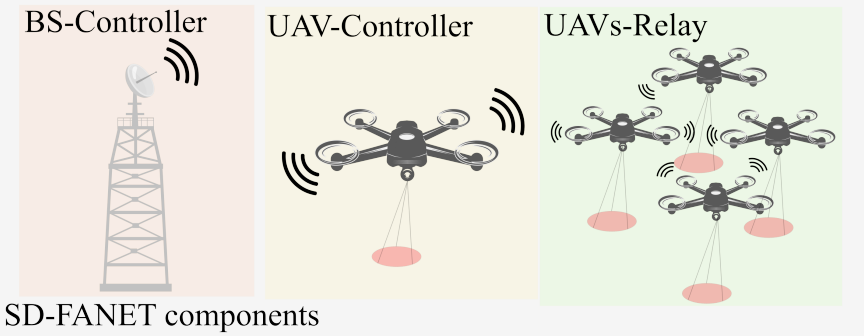

Figure 2. Representation of SD-FANET components: BS-Controller, on the ground, UAV-Controller and UAVs-Relay, on the air, which composes the UAV network.

The UAV-Controller acts as a local controller of the aircraft fleet communication network where it is located. It has the responsibility for the local domain, decentralizing the network control of the BS-Controller. Finally, the other nodes, called UAV-Relay, act as switches capable of interacting with a controller and forward data by consulting its flow table. Table 3 presents the components of the SD-FANET architecture and summarizes their respective functions.

Table 3. Components of the SD-FANET architecture and their main functions.

Component	Function
BS-Controller	Maintain a global view of the network
	Promote UAV-Controller
	Coordinate UAV-Controller
	Create traffic priority policies
	Failure Handling
UAV-Controller	Keep local view of the network
	Coordinate UAV-Relay
	Execute instructions received from BS-Controller
	Create traffic priority policies
	Failure Handling
UAV-Relay	Forward data according to the flow table

3.2. Resilient Strategy

There are at least two basic strategies that a distributed system can adapt to failure. The first one is to use a system capable of isolating the fault and keeping the other components working continuously. The second alternative is to temporarily interrupt the system operation and take some time to reorganize it. During the reorganization period, the status of system components can be evaluated, pending tasks can be completed or dropped, and the crash recovery routine can be triggered. The system reorganization process should be managed, preferably, by a single node called coordinator, which must be selected among the active nodes through an election process[Garcia-Molina 1982].

The SD-FANET architecture uses the second option, which temporarily interrupts the operation of the network to re-establish the operation of the control plane. The time

required to execute this strategy is called recovery time. The mechanism is implemented in all nodes of the network and is executed when a node detects UAV-Controller failure. The recovery process is organized in three stages: 1) Detection of the UAV-Controller failure; 2) Election process for new UAV-Controller; and 3) Reestablishment of the SDN controller. Following, we will explain each of these steps.

3.2.1. Fault Detection

The UAV-Controller receives periodic messages from each UAV-Relay registered in its SDN domain. These messages, called *keepAlive*, informs the status of the UAV-Relay and the UAV-Controller, and are responsible for transporting information that feeds the controller statistics, such as battery status, position (GPS coordinates), and communication links with aircraft neighbors. The sending interval between each *keepAlive* message is configurable. By default, it was set to be 4.5 seconds.

Thus, if a UAV-Relay sends a *keepAlive* message and does not get a response from the UAV-Controller within the configured time interval, a failure detection occurs, and the election process for a new UAV-Controller starts. Similarly, if a UAV-Relay does not send a *keepAlive* message to the UAV-Controller within this interval, its state is assigned to be inactive and the aircraft is not considered by the controller's decision making. This work will not address resilience to the data plane.

3.2.2. Election Procedure

The Election Procedure starts immediately after detecting a failure in the UAV-Controller. The UAV-Relay that detected the communication failure with the UAV-Controller will assume the role of conducting the election and send broadcast an *election* message to the network. This information carries the exact datetime the failure was detected and the identification (ID) number of the issuing node. The datetime value is a timestamp that ensures that other nodes in the network only respond to the election process started first. Only the UAV-Relays that have an ID number lower than the one received in the election message will respond. The node conducting the election will choose the one with the lowest ID and publicize the winner at the end of the process. The disclosure of the new UAV-Controller is done through the *coordinator* message, which is also broadcast on the network. This election procedure was based on the Bullying algorithm [Sathesh 2015].

Three assumptions are taken into consideration when electing the new UAV-Controller: there will be no failures in the communication links, the nodes will always remain active during the entire process, and the clocks of the systems embedded in the aircraft must be synchronized for the correct timing of events. Thus, if we guarantee these assumptions, it is also possible to ensure that a single UAV-Controller will be elected after the conclusion of any election process.

3.2.3. Controller Recovery

The Controller Recovery stage starts after sending a *coordinator* message with the chosen UAV-Relay ID. Each UAV-Relay checks if its ID number is the same as the one carried by the *coordinator* message. If so, it sends a new broadcast through the response *coordinator* message to notify the other nodes that it is active and now waits for the *network-info* message from BS-Controller with the information of the nodes registered in the domain that the new UAV-Controller will be responsible. After retrieving the network state information, the new UAV-Control assumes the active state. An example of the sequence

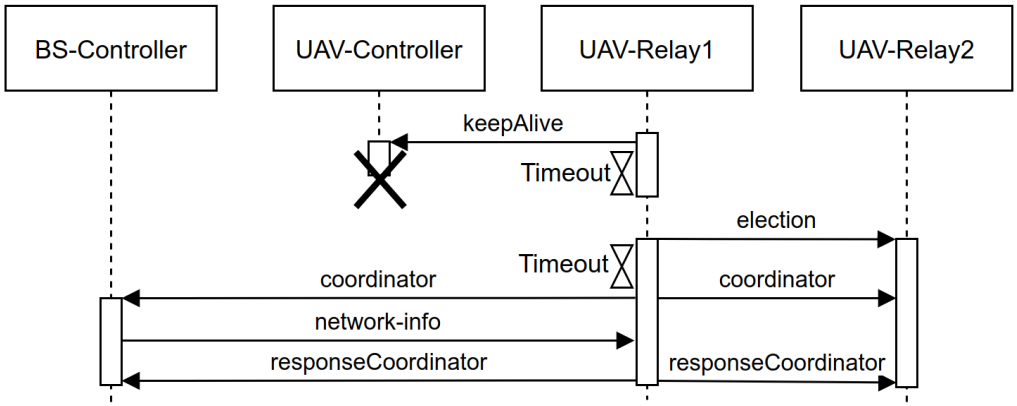

Figure 3. Sequence diagram for the proposed resilient strategy.

diagram with the messages exchanged during the performance of the resilience process is presented in Figure 3.

In this example, UAV-Relay1 sends a *keepAlive* message to UAV-Controller. For any unexpected reason, the UAV-Controller fails. So, UAV-Relay1 detects a timeout for the *keepAlive* message. After that, UAV-Relay1 sends an *election* message and waits for a new timeout. Since there is no UAV-Relay with an ID smaller than UAV-Relay1, no other UAV-Relay sends a response. UAV-Relay1 sends a *coordinator* message to propagate the ID from the new UAV-Controller (which will be UAV-Relay1) when the election timeout is finished. So, BS-Controller sends a *network-info* message to UAV-Relay1 and the new UAV-Controller is activated.

The Algorithm 1 presents the SDN controller's resilience strategy in a simplified way. The method has as input the boolean *keepAliveTimeout*. If the *keepAliveTimeout* condition is true, the election procedure starts and returns the ID of the new UAV-Controller. Then the UAV-Relay is promoted to UAV-Controller and receives its control plane information from BS-Controller.

4. METHODOLOGY

The proposed resilience strategy was evaluated considering the recovery time in an experimental setting. Besides, we also performed an analysis of the configuration *keepAlive* timeout interference in the control plane performance.

4.1. Experimental Setup

We used three DJI Phantom 3 Standard drones for the construction of the experimental scenario. Each drone has embedded a Raspberry Pi 3 Model B, Digi XBee S2C, and power bank to supply the Raspberry. The XBee S2C module connects to the Raspberry board via USB. A similar hardware architecture was validated in our previous work [Santos et al. 2019]. However, in this work, we used ZigBee as the communication protocol to encapsulate SD-FANET messages. This choice was made based on the image transmission system proposed by us in [Pereira et al. 2020]. The aircraft remained in flight side by side at 30 meters high and 20 meters apart them. We used the digi-xbee library (version 1.4.0) to develop the source code, available in Python3.7. The UAVs and the embedded systems are shown in Figure 4. The time recovery was calculated in BS-Controller through log messages created by the SD-FANET applications.

Algorithm 1: UAV-Controller Replacement

Input: $keepAliveTimeout$
Output: $NewUAV - Controller$
if $keepAliveTimeout = True$ **then**
  $nodeState \leftarrow Election$
  $electionStatus \leftarrow True$
  $electionStartTime \leftarrow currentdatetime$
  $send\_broadcast(election(nodeID))$
  while $responseElection$ ***Timer*** **do**
    if $receive\_message(responseElection)$ **then**
      $coordinatorID \leftarrow remoteNodeID$
    else
      $coordinatorID \leftarrow nodeID$
    end
  end
  $send\_broadcast(coordinator(coordinatorID))$
  if $coordinatorID = nodeID$ **then**
    $nodeController \leftarrow True$
    $UAV - Controller(start)$
    $send\_broadcast(responseCoordinator(nodeID))$
    $wait\ network - info\ message$
    $nodeState \leftarrow Active$
  else
    $wait\ responseCoordinator\ message$
  end
else
  $UAV - ControllerState \leftarrow Active$
end

To evaluate our proposed methodology, we built three scenarios, where the difference between them is the *keepAlive* timeout parameter. In Scenarios A, B, and C the *keepAlive* timeout is, respectively, 1.5, 3.0, and 4.5 seconds. As explained in the previous section, this parameter is the time used to identify a UAV-Controller failure and it is also used during the election process to await the candidacy of UAVs-Relay. The smaller the *keepAlive* timeout value, the greater the number of control messages on the UAV network.

We ran 100 tests for each scenario, summing up to 300 fail detection. For comparison purposes, we computed the minimum, maximum, mean, and standard deviation time. Additionally, we will evaluate the efficiency of the strategy for selecting a new UAV-Controller and the influence of the *keepAlive* timeout on the total recovery time.

5. RESULTS

The control plane was reestablished in all tests, demonstrating the efficiency of the strategy. The experimental results for the recovery time are summarized in Table 4 and Table 5 and they are showed in Figure 5. Table 4 details each scenario and their respective

Figure 4. Fleet of UAV and the embedded system used in experimental scenarios.

keepAlive timeout. Scenario A (1.5s timeout) with a minimum time of 3.2179s, maximum time of 3.5816s, mean 3.4266s, and a standard deviation of 0.0622s obtained the best results. However, it should be noted that these values include the election process timeout, which is equal to the *keepAlive* timeout. So, it was expected that the scenario with the lowest *keepAlive* would be faster.

Table 4. Experimental Results to Recovery Time

Scenario	A	B	C
Timeout	1.5	3.0	4.5
Minimum	3.2179	4.7021	6.3619
Maximum	3.5816	5.0805	6.6931
Mean	3.4266	4.9371	6.4648
Standard Deviation	0.0622	0.0653	0.0725

To allow a better analysis of the proposed strategy, Table 5 details the recovery time excluding the election process timeout and thus making a fairer comparison between the scenarios. In this new analysis, scenario B showed the minimum recovery time value between all scenarios. The maximum recovery time was 1.9648s in scenario C. Scenario A is the best case regarding the mean and standard deviation, which are 1.9266 seconds and 0.0622 seconds, respectively. The results of the recovery time considering both with and without the Timeout are illustrated in Figure 5.

Another interesting analysis is that it was possible to compute an overall average for the 300 executions, resulting in an average of 1.9428 seconds with a 0.0685s standard deviation. So, disregarding the constant and configurable time of the election process, the proposed strategy performs the control plane recovery in less than 2 seconds on

Table 5. Experimental Results to Recovery Time Extracting The Election Timeout

Scenario	A	B	C	Total
Minimum	1.7179	1.7021	1.8619	1.7021
Maximum	2.0816	2.0805	2.1931	2.1931
Mean	1.9266	1.9371	1.9648	1.9428
Standard Deviation	0.0622	0.0653	0.0725	0.0685

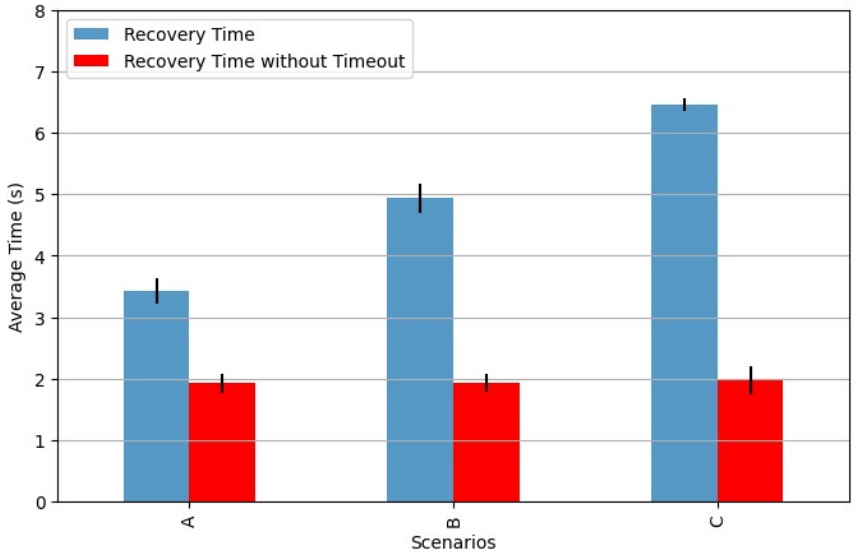

Figure 5. Comparison of results for recovery time with and without timeout.

average. Figure 6 shows a histogram with the number of experiments for some specified time interval, where the majority of the experiments (over 100 samples) are concentrated between 1.90 and 1.95 seconds. Besides, we also plot the probability density function (PDF) of the results, where it is possible to highlight that the behavior of the results is similar to a normal distribution.

To demonstrate the impact of *keepAlive* timeout, Figure 7 shows the number of messages generated for each scenario considering a total time of up to 100 seconds. In this case, Scenario C would generate 132 messages, while scenarios B and A would generate 198 and 396 messages, respectively. Hence, it is necessary to evaluate the amount of control information that will traffic on the network in order to make viable the functioning of the data plane.

6. CONCLUSIONS

Services offered by multi-UAV systems are growing and it is necessary to develop solutions capable of managing the communication network between aircraft. The use of SDN allows for extending and making this management more flexible. Furthermore, resilience strategies must be investigated, as intermittent communication links and the aircraft's limitations require the multi-UAV system to be able to recover from failures.

The solution presented in this paper proved to be efficient for the studied scenarios. In all configurations, the control plane was re-established, where we could achieve an average time of 1.9428 seconds for the control plane recovery. Also, it was possible to

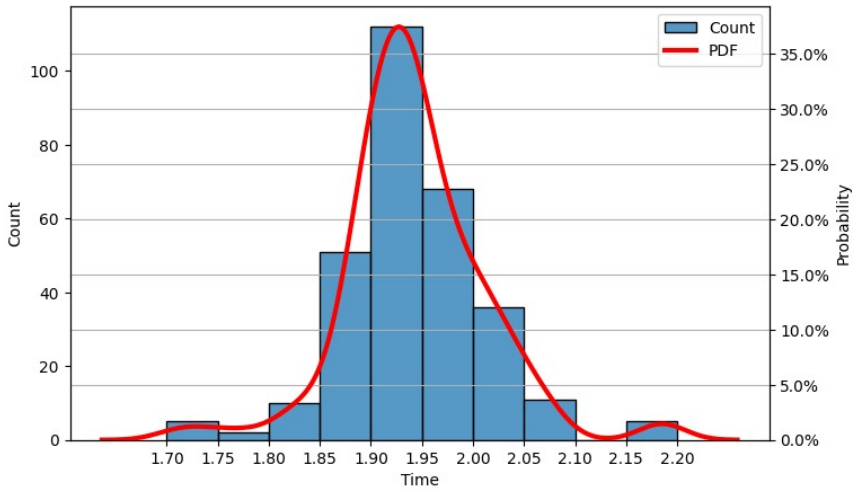

Figure 6. Recovery time behavior of the SD-FANET architecture from the all experimental results.

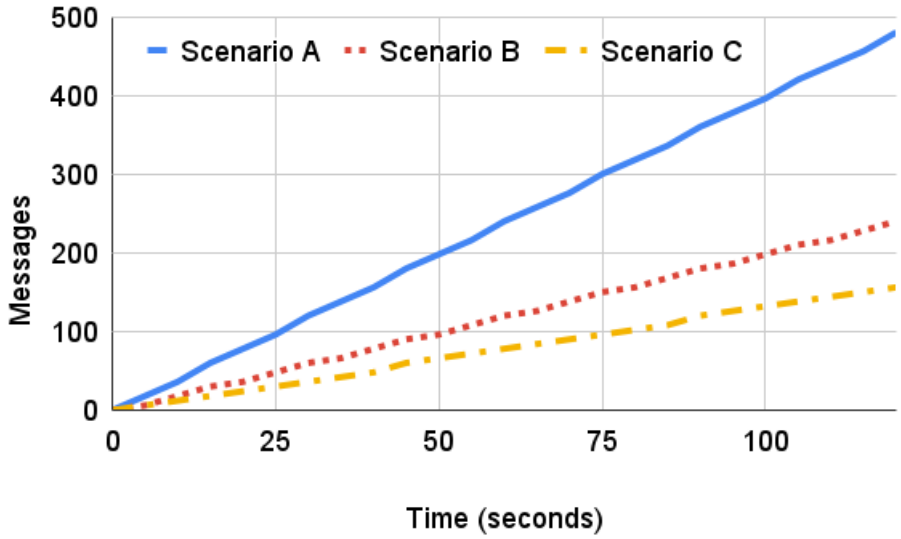

Figure 7. Relationship between the increasing number of messages and selected keepAlive timeout.

observe that the recovery time is more closely related to the *keepAlive* timeout due to the wait time of the election process. When this time was excluded from the analysis, it was possible to notice a similarity between the results of the three proposed scenarios.

Additionally, the reduction of the *keepAlive* sending interval causes overhead in the communication network that can affect the performance of the control plane. Between Scenarios A and C, there was a difference of 300% in the number of messages sent in the same interval.

In future works, we plan to carry out performance evaluations in scenarios with a higher number of nodes, different technologies, and others election algorithms. Also, we will enhance the SDN controller source code for better system performance and try to reduce the recovery time.

Acknowledgment

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001.

References

Alotaibi, E. T., AlQefari, S. S., and Koubaa, A. (2019). Lsar: Multi-uav collaboration for search and rescue missions. *IEEE Access*.

Ammar, H. A., Nasser, Y., and Kayssi, A. (2017). Dynamic sdn controllers-switches mapping for load balancing and controller failure handling. In *2017 International Symposium on Wireless Communication Systems (ISWCS)*, pages 216–221. IEEE.

Bekmezci, I., Sen, I., and Erkalkan, E. (2015). Flying ad hoc networks (fanet) test bed implementation. In *2015 7th International Conference on Recent Advances in Space Technologies (RAST)*, pages 665–668. IEEE.

Binol, H., Bulut, E., Akkaya, K., and Guvenc, I. (2019). Time optimal multi-uav path planning for gathering its data from roadside units. In *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, pages 1–5. IEEE.

Botelho, F., Bessani, A., Ramos, F., and Ferreira, P. (2014a). Smartlight: A practical fault-tolerant sdn controller. *arXiv preprint arXiv:1407.6062*.

Botelho, F., Bessani, A., Ramos, F. M., and Ferreira, P. (2014b). On the design of practical fault-tolerant sdn controllers. In *2014 third European workshop on software defined networks*, pages 73–78. IEEE.

Cumino, P., Lobato Junior, W., Tavares, T., Santos, H., Rosário, D., Cerqueira, E., Villas, L., and Gerla, M. (2018). Cooperative uav scheme for enhancing video transmission and global network energy efficiency. *Sensors*, 18(12):4155.

e Silva, T. D., de Melo, C. F. E., Cumino, P., Rosario, D., Cerqueira, E., and De Freitas, E. P. (2019). Stfanet: Sdn-based topology management for flying ad hoc network. *IEEE Access*, 7:173499–173514.

Foerster, K.-T., Schmid, S., and Vissicchio, S. (2018). Survey of consistent software-defined network updates. *IEEE Communications Surveys & Tutorials*.

Fu, Z., Mao, Y., He, D., Yu, J., and Xie, G. (2019). Secure multi-uav collaborative task allocation. *IEEE Access*, 7:35579–35587.

Garcia-Molina, H. (1982). Elections in a distributed computing system. *IEEE Computer Architecture Letters*, 31(01):48–59.

Gupta, L., Jain, R., and Vaszkun, G. (2015). Survey of important issues in uav communication networks. *IEEE Communications Surveys & Tutorials*, 18(2):1123–1152.

Hu, N., Tian, Z., Sun, Y., Yin, L., Zhao, B., Du, X., and Guizani, N. (2021). Building agile and resilient uav networks based on sdn and blockchain. *IEEE Network*, 35(1):57–63.

Lakhani, G. and Kothari, A. (2020). Coordinator controller election algorithm to provide failsafe through load balancing in distributed sdn control plane. In *International Conference on Computing Science, Communication and Security*, pages 234–250. Springer.

Li, X. and Savkin, A. V. (2021). Networked unmanned aerial vehicles for surveillance and monitoring: A survey. *Future Internet*, 13(7):174.

Liu, C. H., Ma, X., Gao, X., and Tang, J. (2019). Distributed energy-efficient multi-uav navigation for long-term communication coverage by deep reinforcement learning. *IEEE Transactions on Mobile Computing*.

Moazzeni, S., Khayyambashi, M. R., and Movahhedinia, N. (2019). Improving the reliability of software-defined networks with distributed controllers through leader election algorithm and colored petri-net. *Wireless Personal Communications*, 109(1):645–656.

Moazzeni, S., Khayyambashi, M. R., Movahhedinia, N., and Callegati, F. (2018). On reliability improvement of software-defined networks. *Computer Networks*, 133:195–211.

Pereira, D., Nascimento, L., Santos, V., Fernandes, D., and Alsina, P. (2019). Sd-fanet: uma arquitetura para redes aéreas definidas por software aplicadas à varredura de área. In *Anais Estendidos do IX Simpósio Brasileiro de Engenharia de Sistemas Computacionais*, pages 71–76. SBC.

Pereira, D. S., De Morais, M. R., Nascimento, L. B., Alsina, P. J., Santos, V. G., Fernandes, D. H., and Silva, M. R. (2020). Zigbee protocol-based communication network for multi-unmanned aerial vehicle networks. *IEEE Access*, 8:57762–57771.

Qi, W., Song, Q., Kong, X., and Guo, L. (2017). A traffic-differentiated routing algorithm in flying ad hoc sensor networks with sdn cluster controllers. *Journal of the Franklin Institute*.

Santos, V. G., Pereira, D. S., Alsina, P., Fernandes, D. H., Nascimento, L. B., Leite, D. L., Morais, M. R., Silva, M. R., and Souza, E. S. (2019). Multi-uav system architecture for environmental protection area monitoring. In *Proc. Anais do Simpsio Brasileiro de Automao Inteligente*, pages 1–6.

Sathesh, B. (2015). Optimized bully algorithm. *International Journal of Computer Applications*, 121(18).

Tang, Y., Hu, Y., Cui, J., Liao, F., Lao, M., Lin, F., and Teo, R. S. (2019). Vision-aided multi-uav autonomous flocking in gps-denied environment. *IEEE Transactions on Industrial Electronics*, 66(1):616–626.

Vazquez-Carmona, E. V., Vasquez-Gomez, J. I., and Herrera-Lozada, J. C. (2019). Environmental monitoring using embedded systems on uavs. *IEEE Latin America Transactions*, 18(02):303–310.

Yue, X., Liu, Y., Wang, J., Song, H., and Cao, H. (2018). Software defined radio and wireless acoustic networking for amateur drone surveillance. *IEEE Communications Magazine*, 56(4):90–97.

Zhao, Z., Cumino, P., Souza, A., Rosário, D., Braun, T., Cerqueira, E., and Gerla, M. (2019). Software-defined unmanned aerial vehicles networking for video dissemination services. *Ad Hoc Networks*, 83:68–77.

Zhou, Y., Rao, B., and Wang, W. (2020). Uav swarm intelligence: Recent advances and future trends. *IEEE Access*, 8:183856–183878.