CHAPTER 1

INTRODUCTION

Software-Defined Networking (SDN) paradigm has revolution the concept of the traditional networks by decoupling the forwarding plane and control plane making the network open and programmable through a centralized SDN controller. The SDN controller can bethought of as programmable software that can implement any type of application-specific logic (e.g. routing, load-balancing, traffic detouring, VPNs, etc...) translating this high-level logic into low-level rules understandable by forwarding devices. The forwarding devices will simply follow those rules, reacting to particular network events based on the application-specific logic described by them. Various application fields stand to gain significant benefits fromprogrammable networks and the flexibility that they allow. For instance, the integration of Unmanned Aerial Vehicles (UAVs) with SDN can pave the way for breakthrough applications, especially when it comes to emergency or disaster scenarios. Indeed, UAVs are commonly used in such scenarios to dynamically deploy support temporary networks when ground networking infrastructures are heavily damaged or completely torn down. In addition to providing netswork connectivity, UAVs can be used to deliver network security functions like Intrusion Detection Systems (IDSs). Such special UAVs, which is named IDS-enabled UAVs, are meant to safeguard the formed temporary network avoiding that malicious activities or network misuses can damage and disrupt it.

This would be a critical issue since the support network formed by the UAVs is the only way to communicate with rescues. Nevertheless, it is also mandatory to ensure that IDS-enabled UAVs are not burdened too much with the traffic analysis carried out by the IDS function on boarded on them. For such a reason, it is crucial to monitor in a real-time manner the current resource utilization experienced by the IDS-enabled UAVs. This is where the significance of the SDN controller becomes apparent, showcasing its value and importance. Having comprehensive control over the support network, it can be programmed to monitor and retrieve the resource utilization of the IDS-enabled UAVs and promptly act if some of them are particularly burdened

with a huge workload. In pursuing this objective, a load-balancing strategy can be integrated within the SDN controller in order to equally distribute to the special UAVs that onboard the IDSfunction the network traffic that should be analyzed. However, while literature widely addressed the load-balancing of requests in clusters of servers, there is a notable gap in applying such methods to balance the load imposed on UAVs carrying out network security functions, like IDS functions. This gap, especially in critical disaster scenarios, must be filled in order to improve theresilience and responsiveness of such special UAVs when they undergo increasing or hugetraffic loads while also allowing for efficient resource utilization. With the aim of filling this research gap, As extended our prior work proposing an SDN FANET deployment framed in a disaster scenario. The SDN controller manages the UAVs forming the FANET, constantlyretrieving status information about them and about the network. Some of these UAVs, other than providing connectivity, execute an additional IDS function to classify network flows that traverse the SDN-FANET, determining whether a malicious activity is taking place or not. In thisframework, exploiting the SDN controller programmability, The proposed a new Software-defined UAV Resource Aware Load-Balancing Strategy LB to fairly distribute the network flows to be analyzed among the IDS-enabled UAVs through a real-time and comprehensive analysis of their resource utilization and the status of the network.

The proposed LB strategy is compared against baseline strategies, a naive one such as Random which does not follow a specific rule to distribute the workload, but distributes it in a random way; and Round Robin, a more intelligent load-balancing strategy also considered in as a valuable benchmark. However, none of them takes into consideration the UAVs' and network resource status, i.e. they can be defined as resource-agnostic strategies. The results show that the proposed load balancing strategy outperforms the Random and Round-Robin ones, demonstrating a more efficient and intelligent workload distribution among the IDS-enabled UAVs due to its resource-aware feature.

1.1 IDS over FANET

Some of the papers in the literature that deal with IDS mechanisms and Machine-learning approaches over a network of UAVs are described in this section. A brief survey is presented in [8] where the authors show the recent state-of-the-art on this topic providing a classification of the different IDS mechanisms and a taxonomy of the UAV-IDS system components. Moreover, they provide a view of the research challenges, insights, and future research directions. In the Principal Component Analysis (PCA) is used together with one-class classifiers as mechanismsto detect attacks. Their approach considers an IDS that is onboard a UAV and uses a resource- constrained agent for detecting and eventually mitigating attacks even when communication to the ground control station is lost from jamming. However, the proposal does not consider the computational overhead and the additional workload imposed on the UAV that executes the model to classify the network traffic. This additional workload can highly harm the resources of the UAV.

Therefore, it is crucial a resource-aware and real-time monitor of the status of this special UAV and, if needed, take actions to reduce that computational burden imposed on them to preserve their resources. In the authors consider a Deep Learning (DL) approach in the development of an IDS. They introduce a real time data analytics platform based on DL to look into FANET intrusion detection risks. A stream processing module in this architecture collects communication from UAV, including data on intrusion detection. In the authors introduced an IDS based on spectral traffic analysis and a reliable controller for anomaly estimation. This method was developed to address distributed denial-of-service (DDoS) attacks. The outcomes demonstrated precise identification of several anomaly kinds using real-time traffic. Nevertheless, in the previous works, the IDS has not been on boarded on the UAVs and used it tobolster their real time classification capabilities. Rather, its utilization lies in classifying network traffic collected by the UAV by means of the centralized entity, like the SDN controller,potentially resulting in successful attacks going unnoticed or in a delayed detection. Finally, noneof the mentioned works fully exploit the modern programmable features enabled by the SDN paradigm to obtain more flexible and fine-grained management of the UAVs forming the FANET.

1.2 IDS over SDN FANET

The join between Machine Learning and SDN is analyzed by the authors that provide a methodology to identify two main types of attack in a network of drones: intrusion from the outside and network usage from inside. In [13] the author proposes an intrusion detection model based on the attention mechanism. They use two different mechanisms, deep auto-encoder(DAE) for reducing invalid features to improve the efficiency of subsequent training and discarding invalid features, and convolutional neural network (CNN) for extracting the abstract features, then used to detect and classify the traffic. However, none of the other proposals leverage the integration between IDS and a load-balancing strategy for a fair distribution of the network flows in an SDN context.

The seamless integration between SDN and FANET enables real-time monitoring and data retrieval concerning both the network and UAVs. This data can then be used to feed a load balancing module, embedded within the SDN controller, that could help in quickly taking actions to reduce the computational burden that is imposed on the UAVs that, other than providing connectivity, execute additional functionalities, like an IDS function.

This is paramount to ensure that the lifetime of the UAVs is not highly affected, thus prevent ing them from being abruptly torn down and causing malfunctioning to the network. This scenario is further exacerbated when dealing with disaster events due to the critical nature that requires a resource-aware, prompt, and effective solution.

1.3 Scope of the project

This project focuses on developing a Software Defined Intrusion Detection System (IDS) integrated with UAV Resource Aware Load Balancing for managing disaster scenarios in Flying Ad-hoc Networks (FANETs). The scope includes defining FANET disaster scenarios and identifying network challenges such as instability and resource constraints. A comprehensive literature review will guide the design of a modular system architecture that integrates SDN principles for dynamic network management, real-time IDS for threat detection, and UAVresource management for efficient load balancing. Development will encompass creating an

SDN controller with OpenFlow support, implementing IDS algorithms for anomaly detection, and devising UAV resource management strategies. Testing will involve simulations and real-world experiments to evaluate IDS effectiveness, network performance metrics (throughput, latency), and UAV resource utilization. Security and privacy considerations will ensure secure communication and data privacy.

1.5 Problem Statement

In the context of a disaster scenario, where rapid and efficient communication and data management are crucial, Software Defined Intrusion Detection Systems (IDS) coupled with UAV (Unmanned Aerial Vehicle) resource-aware load balancing present a novel solution for maintaining network integrity and reliability. The problem statement revolves around the challenge of dynamically managing and securing an ad-hoc network that often experiences unpredictable loads and resource constraints during emergencies. Traditional IDS may not scale effectively in such dynamic environments, leading to potential security gaps. By leveraging UAVs equipped with software-defined IDS, the system aims to provide adaptive and scalable network monitoring while intelligently balancing the load across available resources. This approach not only enhances the network's resilience against intrusions but also ensures optimal performance under varying conditions, thus addressing the critical needs of disaster response operations.

In disaster scenarios, Software Defined IDS with UAVs enables dynamic and scalable network security by using UAVs to monitor and manage network traffic. Resource-aware load balancing ensures efficient distribution of network tasks among UAVs, enhancing both security and performance in highly variable environments. This approach addresses the unique challenges of disaster response networks by adapting to changing conditions and ensuring robust protection and connectivity. solution for this Develop a software-defined IDS with UAV resource-aware load balancing in FANETs to optimize security and efficiency in disaster response scenarios.

In Chapter 2 Let us discuss on literature survey on software-defined intrusion detection systems (IDS) with UAV resource-aware load balancing in FANET disaster scenarios highlights the integration of SDN for dynamic and efficient network management and IDS for enhanced security, ensuring optimal resource utilization and resilient communication during emergency response.

In Chapter 3 Let us discuss on Software Requirements Specification (SRS) is a comprehensive document that outlines the functional and non-functional requirements, system behavior, and constraints for a software project, serving as a blueprint for development and validation.

In Chapter 4 Let us discuss on System analysis involves examining and evaluating the components and processes of an existing or proposed system to understand its functionality, identify improvements, and ensure it meets user requirements and business objectives.

In Chapter 5 Let us discuss on System design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements and ensure effective system.

In chapter 6 Let us discuss about Implementation is the process of executing the system design by developing, integrating, and deploying the software or system components to meet the specified requirements and ensure operational functionality.

In Chapter 7 Let us discuss on Testing is the process of evaluating a system or its components to verify that it meets the specified requirements and to identify any defects or issues before deployment.

In chapter 8 Let us discuss about Results are the outcomes or data obtained from testing and evaluation processes, demonstrating whether the system meets the specified requirements and performance.

CHAPTER 2

LITERATURE SURVEY

The literature survey reveals that current FANET security solutions are inadequate for disaster scenarios, necessitating a software-defined intrusion detection system (IDS) with UAV resource- aware load balancing. Existing solutions focus on either IDS or load balancing, neglecting the interdependence of security and resource allocation. Software-defined networking enables dynamic security configuration, while UAV resource-aware load balancing optimizes FANET performance. Integrated IDS and load balancing solutions can enhance FANET resilience and security.

Researchers propose various IDS techniques, including machine learning and anomaly detection, but these require integration with load balancing considerations. The survey highlights the importance of considering UAV resources, disaster scenario constraints, and FANET dynamics. A software-defined IDS with UAV resource-aware load balancing can detect and prevent intrusions while ensuring efficient FANET operation. Key challenges include developing adaptive security measures, optimizing resource allocation, and integrating IDS and load balancing. Further research is needed to design and evaluate a comprehensive system thataddresses these challenges and ensures reliable FANET operation in disaster scenarios. Bybridging the gap between security and resource allocation, such a system can significantly enhance FANET resilience and effectiveness.

Existing solutions focus on either IDS or load balancing, but not both. Software-defined networking can enable dynamic security configuration, while UAV resource-aware load balancing is crucial for efficient FANET operation. Integrated IDS and load balancing solutions can enhance FANET resilience and security. Researchers propose various IDS techniques, including machine learning and anomaly detection, but these need to be integrated with load balancing considerations. The survey underscores the importance of considering UAV resources and disaster scenario constraints in designing a comprehensive solution.

A software-defined IDS with UAV resource-aware load balancing can effectively detect and prevent intrusions while ensuring efficient FANET operation in disaster scenarios. Further research is needed to develop and evaluate such a system.

2.1 Related works

Many studies in the literature provide examples of UAVs used to create rapid connectivity coverage, but only some works make use of IDS technologies for providing security over these networks, and no papers exist taking into account together IDS, load balancing, and SDN technologies in an emergency scenario.

Related works on software-defined intrusion detection systems and UAV resource-aware load balancing for FANETs in disaster scenarios are diverse. Machine learning-based IDS for FANETs, anomaly detection, and UAV resource-aware load balancing are explored. SDN-basedFANET architecture and security solutions are also investigated. Integrated IDS and load balancing solutions are proposed, considering FANET security in disaster scenarios and UAV resource allocation. Research highlights the importance of dynamic security configuration, resource allocation, and UAV resource awareness. Works also emphasize the need for adaptive security measures, optimized resource allocation, and integrated IDS and load balancing. These related works lay the foundation for designing a software-defined IDS with UAV resource-awareload balancing to enhance FANET security and resilience in disaster scenarios. By building on these studies, a comprehensive system can be developed to address the unique challenges of FANETs in disaster scenarios.

2.2 Main contributions

In light of the considerations made so far and extending our prior work [6], the main contributions to the advancement of the state of the art that our work aims to provide are the following:

• An SDN-FANET deployment in disaster scenarios enhanced with UAVs that act as IDS exploiting a deep learning approach to classify network flows identifying whether malicious activities are taking place or not.

• A UAV Resource Aware (URA) Load-Balancing strategy LB integrated within the SDN controller which helps in fairly distributing network flows to be analyzed among the IDS- enabled UAVs through a real-time and comprehensive analysis of their resource utilization.

• A dynamic approach that, by monitoring the resources of such IDS-enabled UAVs, promptly activates additional IDS-enabled UAVs within the SDN-FANET to react and adapt the safeguarding task to the increasing traffic loads.

• The main entities of the proposed framework and their interactions are described in the System Model subsection along with a graphical representation depicted in Fig. 1.

• In addition, the adopted UAVs' Energy Model formulation is presented also describing the methodology that allows us to adapt it to the considered scenario.

• System model The main components of the proposal are described as follows:

• A set of UAVs $U = \{u1,2,\ldots,un\}$ deployed over an area in order to provide networking capabilities. Each UAV $ui \in U$, $with\ i \in [1, n]$, for such a reason, acts as an OpenFlow programmable Open vSwitch (OvS) managed by an SDN controller.

• We denoted with $|U| = n$ the cardinality of the set U. Therefore, the connectivity service for the area will be provided by means of n UAVs.

• Let F be the universe of all the possible flow. Given a flow $f \in F$, we considered a function σ, named Network Features Extraction (NFE) function, defined as follows:

$\sigma : F \rightarrow \Gamma$ (1)

where Γ is the set of all the possible network features that can be extracted from f using the function σ. The set Γ is finite and fixed by the design of the function σ, depending on the number of features that it can compute over a flow f.

• We denoted with ρ an IDS function defined as follows:

$\rho : \Gamma \rightarrow D$, $with\ D$={0,1} $\rho(\sigma(f))$ = (2) More specifically, given a flow $f \in F$ the IDS function ρ applied to f, denoted as $\rho(\sigma(f))$, is such that: { 0, $if\ f\ classified\ as\ benign$ 1, $if\ f\ classified\ as\ malicious$

In our case, the IDS function is implemented as a Deep Neural Network (DNN) that given the network features extracted from a flow $f \in F$ is able to classify it as benign or not. More specifically, given the set of the features extracted using f, (f) = X, the function ρ has the following structure: $\rho(X) = W \cdot X + B$ (3)

Summary:

The literature survey reveals that software-defined IDS for FANETs in disaster scenarios leverages UAVs for dynamic, real-time intrusion detection and resource management. Key challenges include ensuring scalable, real-time processing and balancing load based on UAV resource constraints. Studies emphasize the need for adaptive load-balancing algorithms that consider UAV capabilities and task priorities. Innovations focus on integrating machine learning for improved detection and optimizing energy use. Future research should enhance real-world testing and interoperability for practical implementation in disaster-stricken environments.

CHAPTER 3

SYSTEM REQUIREMENTS AND SPECIFICATION

The project requires an SDN controller to manage FANETs, integrating a real-time IDS for threat detection and response. It includes a UAV resource management module to optimize resource allocation, ensuring efficient load balancing. Security measures are essential for secure UAV communication and data integrity. The system will be validated through simulations and real-world tests, with comprehensive documentation covering system architecture, implementation details, and performance evaluations for future enhancements.

3.1 Functional Requirements

Detecting intrusions in Flying Ad Hoc Networks (FANETs) involves several functional requirements to ensure the network's security and reliability. Here's a detailed breakdown of what these requirements might include:

3.1.1 Real-Time Monitoring and Detection

- **Continuous Surveillance**: The system should constantly monitor network traffic and node behaviors to identify potential security threats or anomalies in real-time.
- **Immediate Alerts**: Upon detecting suspicious activities, the system should generate alerts for network administrators or automated responses to address the threat quickly.

3.1.2. Anomaly Detection

- **Behavioural Analysis**: Establish baselines of normal behavior for each node and thenetwork as a whole, then identify deviations from these norms.
- **Traffic Analysis**: Monitor data packets for unusual patterns, such as unexpected traffic volume, unauthorized data exchanges, or irregular communication patterns.

Identifying anomalies in network traffic is a crucial aspect of maintaining security in any network, including Flying Ad Hoc Networks (FANETs). Here's a comprehensive breakdown of the functional requirements for effectively identifying anomalies in network traffic:

3.1.3. Traffic Collection and Analysis

- **Data Capture**: Implement mechanisms to capture all relevant network traffic, including packet headers and payloads, across the FANET.
- **Traffic Filtering**: Use filtering techniques to focus on specific types of traffic or particular nodes, if necessary, to reduce noise and improve analysis efficiency.

3.1.4. Anomaly Detection Techniques

- **Statistical Methods**: Apply statistical models to establish baseline traffic patterns and identify deviations from these patterns. Techniques include mean, variance, and standard deviation calculations.
- **Machine Learning**: Utilize machine learning algorithms to detect anomalies. Supervised learning methods (e.g., classification) and unsupervised learning methods (e.g., clustering) can be employed to identify patterns and outliers.
- **Behavioral Analysis**: Monitor and analyze node behavior over time to detect deviations from typical behavior. This can include unusual communication frequencies, unexpected data flows, or abnormal connection patterns.
- **Signature-based Detection**: Compare traffic patterns against known attack signatures to identify potential threats.

3.1.5. Anomaly Characterization

- **Type Identification**: Classify detected anomalies based on their nature, such as traffic spikes, unusual packet types, or deviations from expected communication protocols.
- **Severity Assessment**: Assess the severity of anomalies to prioritize response actions. Determine if the anomaly indicates a potential attack, misconfiguration, or benign issue.

3.1.6. Alerting and Reporting

- **Real-time Alerts**: Generate immediate alerts for detected anomalies to allow quick investigation and response. Include details such as the type of anomaly, affected nodes, and potential impacts.
- **Incident Reporting**: Provide detailed reports on anomalies, including historical context,

- patterns, and potential causes, for further analysis and documentation.

3.1.7. Integration with Other Security Measures

- **Correlation with Other Data**: Integrate anomaly detection with other security systems (e.g., firewalls, intrusion prevention systems) to cross-reference and validate findings.
- **Feedback Loop**: Incorporate feedback from incident investigations to refine and improve the anomaly detection algorithms and rules.

3.1.8. Performance and Scalability

- **Efficiency**: Ensure that anomaly detection processes do not introduce significant overhead or degrade the performance of the FANET nodes.
- **Scalability**: Design the system to handle varying sizes and dynamic changes in the FANET, including the addition or removal of nodes and changes in network topology.

3.1.9. User Interaction and Configuration

- **Customizable Thresholds**: Allow users to set and adjust thresholds for anomaly detection to tailor the system to specific network requirements and conditions.
- **User Interface**: Provide a user-friendly interface for monitoring detected anomalies, configuring detection parameters, and managing alerts.

3.1.10. Continuous Improvement

- **Learning and Adaptation**: Implement mechanisms for the system to learn from past anomalies and adapt its detection techniques accordingly.
- **Regular Updates**: Update detection algorithms and signatures regularly to address emerging threats and evolving network conditions.

3.1.11. System Reliability and Security

- **Redundancy**: Ensure the anomaly detection system has redundancy and failover mechanisms to maintain functionality in case of system failures.

- **Secure Access**: Protect the anomaly detection system from unauthorized access and tampering to ensure the integrity of the detection processes.
- By addressing these requirements, you can develop a robust system for identifying anomalies in network traffic that helps maintain the security and efficiency of FANETs.

3.2 Non-Functional Requirements

Non-functional requirements for real-time processing and detection in the context of network security, such as for Flying Ad Hoc Networks (FANETs), focus on how well the system performs its functions under specific conditions. Here's a detailed breakdown of non- functional requirements related to real-time processing and detection:

3.2.1. Performance

- **Latency**: The system must process and detect anomalies with minimal delay. Specifically, the latency from data capture to detection should be kept under a predefined threshold (e.g., milliseconds or seconds) to ensure timely identification of potential threats.
- **Throughput**: The system should be capable of handling high volumes of network traffic without degradation in performance. It should process a large number of packets persecond, depending on the network scale and expected traffic load.

3.2.2. Scalability

- **Horizontal Scalability**: The system should scale horizontally to accommodate increases in network size and traffic volume. This may involve adding more processing nodes orresources to handle increased loads.
- **Dynamic Adaptation**: The system should dynamically adapt to changes in network topology and traffic patterns. This includes handling the addition or removal of nodes and adjusting processing resources as needed.

3.2.3. Reliability

- **High Availability**: The system should be designed for high availability to ensure continuous operation. This may involve redundant components, failover mechanisms, and distributed processing to minimize downtime.

- **Error Handling**: Robust error detection and handling mechanisms must be in place to ensure that anomalies do not lead to system failures or data loss.

3.2.4. Efficiency

- **Resource Utilization**: The system should optimize the use of computational resources (CPU, memory, and storage) to maintain efficient operation without excessive consumption.
- **Energy Consumption**: In the context of FANETs, where nodes may be battery-powered, the system should be designed to minimize energy consumption during anomaly detection processes.

3.2.5. Accuracy

- **Detection Accuracy**: The system should accurately identify true positives and minimize false positives and false negatives. High detection accuracy ensures that real threats are identified while reducing unnecessary alerts.
- **Adaptability**: The detection algorithms should adapt to evolving traffic patterns and emerging threats to maintain high accuracy over time.

3.2.6. Security

- **Data Privacy**: Ensure that network traffic data is protected and anonymized if necessary to preserve user privacy. Encryption should be used for data in transit and at rest.
- **Access Control**: Implement strong access controls to protect the system from unauthorized access and manipulation. This includes secure authentication and authorization mechanisms.

3.2.7. Usability

- **User Interface**: Provide an intuitive and user-friendly interface for monitoring and managing real-time detection. The interface should facilitate quick interpretation of alerts and system status.
- **Configurability**: Allow users to easily configure detection thresholds, response actions, and other parameters relevant to real-time processing.

3.2.8. Maintainability

- **Modularity**: Design the system in a modular way to facilitate easy updates and maintenance. Components should be loosely coupled to simplify modifications and enhancements.

- **Documentation**: Provide comprehensive documentation for system configuration, operation, and troubleshooting to support maintenance and user training.

3.2.9. Compliance

- **Standards and Regulations**: Ensure that the system complies with relevant industry standards and regulations related to real-time processing and security, such as data protection laws and network security standards.

3.2.10. Integration

- **Interoperability**: The system should seamlessly integrate with existing network management and security tools. It should support standard communication protocols and APIs for data exchange and coordination.
- **Feedback Mechanism**: Implement a feedback loop to continuously improve the system based on operational experience and user input.

3.2.11. Testing and Validation

- **Continuous Testing**: Implement ongoing testing procedures to validate the performance and accuracy of real-time processing and detection. This includes unit tests, integrationtests, and performance benchmarks.
- **Real-world Simulation**: Conduct tests under realistic conditions to ensure that the system performs effectively in the actual network environment.
- By addressing these non-functional requirements, you can ensure that the real-time processing and detection system for FANETs is efficient, reliable, and capable of meeting.

3.3 System and energy model

In this section, the main entities of the proposed framework and their interactions are described in the System Model subsection along with a graphical representation depicted in Fig. 1. In addition, the adopted UAVs' Energy Model formulation is presented also describing the methodology that allows us to adapt it to the considered scenario.

3.3.1. System model

The main components of the proposal are described as follows:

• A set of UAVs$U = \{u1,2,\ldots,un\}$ deployed over an area in order to provide networking capabilities. Each UAV $ui \in U$, $with\ i \in [1, n]$, for such a reason, acts as an OpenFlow

• We denoted with $|U| = n$ the cardinality of the set U. Therefore, the connectivity service for the area will be provided by means of n UAVs.

• Let F be the universe of all the possible flow. Given a flow $f \in F$, we considered a function σ, named Network Features Extraction (NFE) function, defined as follows:

$$\sigma : F \rightarrow \Gamma (4)$$

where Γ is the set of all the possible network features that can be extracted from f using the function σ. The set Γ is finite and fixed by the design of the function σ, depending on the number of features that it can compute over a flow f.

• We denoted with ρ an IDS function defined as follows:

$$\rho : \Gamma \rightarrow D, with\ D=\{0,1\} \quad (5)$$

More specifically, given a flow $f \in F$ the IDS function ρ applied to f, denoted as $((f))$, is such that

A flow $f \in F$ is univocally identified by means of the five-tuple. It is used to group all the network packets that belong to f and that, therefore, forms the flow f. A five-tuple associated with a flow f is defined as $ftf = (src\_ipf, \_ipf, src\_portf, dst\_portf, protocolf)$. Given the universe of all the possible five-tuple FT, and given the universe of all possible flows F, the five-tuple function ϕ can be defined as follows:

$$\rho(\sigma(f)) = \begin{cases} 0, & if\ f\ classified\ as\ benign \\ 1, & if\ f\ classified\ as\ malicious \end{cases}$$

In our case, the IDS function is implemented as a Deep Neural Network (DNN) that given the network features extracted from a flow $f \in F$ isable to classify it as benign or not. More specifically, given the set of the features extracted using f, $(f) = X$, the function ρ has the following structure:

$X) = W \cdot X +$ B (7)(

where W and B represent the vector of weights and the vector of bias of the DNN, respectively

A flow $f \in F$ is univocally identified by means of the five-tuple. It is used to group all the network packets that belong to f and that, therefore, forms the flow f. A five-tuple associated with a flow f is defined as $ftf = (src\_ipf, \_ipf, src\_portf, dst\_portf, protocolf)$. Given the universe of all the possible five-tuple FT, and given the universe of all possible flows F, the five-tuple function ϕ can be defined as follows:

$\forall f1,2 \in F, f1 \neq f2, \phi(f1) = ftf1, \phi(f2) = ftf2\ s.t.\ ftf1 \neq ft$ (8)

Starting from the tuple, given $f \in F$, it can always be determined a unique identifier of this flow (fid) as follows:

$fid = Hash(\phi(f))\ where$

$\phi(f) = ftf = (src\_ipf, dst\_ipf, src\_portf, dst\_portf, protocolf) \in FT$ (9)

• A set of IDS-enabled UAVs $UIDS = \{u1,2,\ldots,um\}$ defined such that $UIDS \subseteq U$ with $m < n$. This set represents the m UAVs that onboard the IDS function ρ (green UAVs in Fig. 1). More specifically, each $ui \in UIDS$ is associated with

• We denoted with $UFWD$ the set of UAVs that act only as forward ing elements, such that:

• $\cup\ U = UFWD\ UIDS$(11)

The UAVs in $UFWD$ are on boarded with only the OvS function denoted with the symbol φ.

• It is worth noticing that also the UAVs in the set $UIDS$ onboard the φ providing network connectivity, but in addition to it they execute the IDS function ρ.

• The UAVs in $UIDS$ are further divided into two subsets, $UA\ IDS$ and $UI\ IDS$ such that:

$$UIDS = UA \cup UI\ IDS\ and, UA\ IDS \cap UI\ IDS = \emptyset\ |UA\ IDS| = z\ ,|UI\ IDS| = y\ s.t.\ z + y = m \quad (12)$$

3.2 System Model

3.2.1 Network Architecture

- **UAV Nodes**: UAVs act as nodes in the FANET, providing communication, surveillance, and potentially disaster relief functions. Each UAV can function as a data collector, analyzer, or relay.
- **IDS Nodes**: Some UAVs are designated as IDS nodes responsible for monitoring network traffic, detecting anomalies, and managing security alerts.
- **Control Center**: A central or distributed control center coordinates the IDS activities, provides overall network management, and performs higher-level analysis and decision-making.

3.2.3 IDS Components

- **Data Collection**: UAVs equipped with sensors and communication modules capture network traffic and environmental data.
- **Traffic Analysis**: IDS nodes process collected data in real-time using anomaly detection techniques and machine learning algorithms.
- **Alert Generation**: When an anomaly is detected, IDS nodes generate alerts and trigger predefined responses or escalate to the control center.
- **Load Balancing**: Distribute IDS tasks across multiple UAVs based on their resource availability, such as processing power, memory, and energy levels.

3.2.4 Load Balancing Mechanism

- **Task Assignment**: Dynamically assign IDS tasks (e.g., data collection, analysis, reporting) to UAVs based on their current resource availability and workload.
- **Resource Awareness**: Consider UAVs' energy levels, processing capabilities, and communication ranges when assigning tasks.
- **Adaptive Balancing**: Continuously monitor UAV resource status and reallocate tasks asneeded to ensure balanced load and avoid overloading any single UAV

3.3 Energy model

The total energy consumption of a UAV can be modeled taking into account three main contributions as expressed by the following formula:

$E = EENGs + ENICs + EFUNs$

3.3.1 $EENGs$: the energy consumed by the electrical engines, it indicates the vertical energy used by each UAV for reaching a fixed altitude in the sky and that used for moving horizontally following the prefixed UAV mobility model.

3.3.2 $ENICs$: the energy consumed by the network interface cards (NICs), it represents the energy due to the transmission and reception of data on each communication link.

3.3.3 $EFUN$: the energy consumed by the additional functions carried out by the drone

3.4 Energy Model

3.4.1 Energy Consumption Factors

- **Data Collection**: Energy is consumed by sensors and communication modules while collecting and transmitting network traffic data
- **Data Processing**: Energy is required for processing traffic data and running anomaly detection algorithms on the UAVs.
- **Communication Overhead**: Energy is used for communication between UAVs and between UAVs and the control center.
- **Movement and Navigation**: UAVs consume energy during flight, which affects their overall energy availability for IDS tasks.

3.4.2 Energy Management Strategies

- **Energy-Efficient Algorithms**: Implement energy-efficient algorithms for data processing and anomaly detection to minimize energy consumption on UAVs.
- **Task Offloading**: Offload processing tasks to UAVs with higher remaining energy or to those with specialized processing capabilities.

- **Energy Prediction**: Predict UAV energy levels based on current usage patterns and remaining flight time to optimize task assignment and scheduling.
- **Sleep Modes**: Utilize low-power sleep modes for UAVs that are not actively performing IDS tasks, waking them only when needed.

3.4.3 Load Balancing Considerations

- **Load-Resource Mapping**: Map IDS task loads to UAV resources, ensuring that each UAV is assigned tasks proportionate to its available energy and processing capabilities.
- **Energy-Aware Scheduling**: Schedule IDS tasks considering UAV energy levels to prevent premature battery depletion and ensure balanced energy consumption across the network.
- **Redundancy and Failover**: Implement redundancy strategies to handle UAV failures due to energy depletion. Ensure that backup UAVs or nodes can take over tasks seamlessly.
- **Integration and Optimization**

3.4.4 System Integration

- **Inter-UAV Communication**: Ensure seamless communication protocols between UAVs for task coordination, data sharing, and alert propagation.
- **Control Center Coordination**: Integrate UAVs with the control center for centralized monitoring, higher-level decision-making, and coordination of IDS activities.

3.4.5 Optimization Techniques

- **Dynamic Rebalancing**: Continuously monitor UAV performance and resource usage to dynamically rebalance IDS tasks and adjust strategies based on real-time data.
- **Algorithm Tuning**: Regularly tune anomaly detection algorithms and load-balancing strategies to improve efficiency and accuracy based on operational feedback.
- **Simulation and Testing**: Use simulations to test and optimize the system's performance under different disaster scenarios and network conditions.

3.4.6 Disaster Scenario Considerations

- **Adaptability**: Design the system to adapt to rapidly changing conditions typical in disaster scenarios, such as fluctuating network topologies and varying UAV capabilities.
- **Robustness**: Ensure that the system is robust against network disruptions, UAV failures, and other unforeseen challenges during a disaster.
- **Redundancy**: Incorporate redundancy at various levels (e.g., data collection, processing, communication) to enhance system reliability and resilience.

3.5 Hardware Requirements

- **UAVs:** Onboard processors, adequate RAM, high-speed communication modules (4G/5G, Wi-Fi), and sufficient battery life.
- **Ground Control Stations:** High-performance servers with multi-core processors, ample memory/storage, and robust networking infrastructure.
- **Network Infrastructure:** High-bandwidth, low-latency routers and switches, and reliable firewalls.
- **Sensors:** Essential sensors (cameras, GPS) on UAVs for data collection.
- **Power Supply:** Reliable power sources for continuous UAV and GCS operation.

3.6 Software Requirements

- **SDN Controllers:** Software like OpenDaylight or Ryu to manage network flow and device configurations.
- **IDS Software:** Snort or Suricata for real-time intrusion detection and analysis.
- **Load Balancing Algorithms:** Custom or existing algorithms for distributing workloads based on UAV resources.
- **Monitoring Tools:** Tools like Nagios or Zabbix for real-time resource and network performance monitoring.
- **Data Analytics Tools:** Big data tools like Hadoop or Spark for processing and analyzing large volumes of data.

Summary:

The system requirements for a software-defined IDS with UAV resource-aware load-balancing in FANET disaster scenarios include real-time processing capabilities for intrusion detection, dynamic load-balancing to manage UAV resource constraints, and adaptability to fluctuating network conditions. It must support high accuracy in anomaly detection while optimizing energy usage and computational resources. The system should integrate seamlessly withUAVs and network management tools, providing scalability and robustness in disaster environments. Additionally, it requires efficient communication protocols for coordination and response to network.

CHAPTER 4

SYSTEM ANALYSIS

In designing a system for Software-Defined Intrusion Detection Systems (IDS) with UAV Resource-Aware Load Balancing in FANET disaster scenarios, it is essential to consider several critical requirements. The system must ensure robust and adaptive security measures capable of detecting and mitigating threats in real-time, given the dynamic and unpredictable nature of disaster environments. It should incorporate a scalable and flexible architecture to handle varying numbers of UAVs and adapt to changing network topologies. Efficient resource management is paramount, necessitating intelligent load-balancing algorithms that consider the energy constraints and computational capacities of UAVs. The system must also provide seamless and reliable communicationamong UAVs and ground stations, ensuring minimal latency and high data throughput

4.1. Initial IDS activation and initialization phase

At the very beginning, the controller uniformly selects a single UAV $ui \in UIDS$ to be woken up in equation (13)

$$
\begin{aligned}
&i \xleftarrow{R} [1, m] \\
&pick\ u_i \in U_{IDS} \\
&U^A_{IDS} = \{u_i\} \\
&U^I_{IDS} = U_{IDS} \setminus U^A_{IDS}
\end{aligned}
$$

Once selected, the IDS function of ui needs to be initialized. In order to do that, the Initialization Phase is performed. To this aim, a reliable and dedicated communication channel between each IDS-enabled UAV and the SDN controller is established by means of the $SBI2$. The $SBI2$ has been implemented by means of a TCP socket between each IDS enabled UAV and the SDN controller. More specifically, the controller sends to the elected IDS-enabled UAV uian Activation TCP packet with a payload structured as shown in Fig. 2 More specifically, the Activation field is a single bit used to activate or deactivate the IDS function on an IDS-enabled UAV. The following parameters denoted as $W1,2,\ldots,Wk$ and$B1,B2,\ldots,Br$ represent the weights and the biases of the DNN. Finally, the payload contains the DNN template that describes the structure of the DNN (neurons, layers, activation functions and that is initialized using the weights in the payload. Considering a TCP packet, the maximum allowable payload size is

65535 bytes. Therefore both weights and DNN can be transmitted using a single TCP packet since the sum of their size is such to fit this constraint. However, due to the limited resources that typically characterize a UAV, the DNN running on it cannot be too complex in order to avoid resource overhead and quick battery depletion. Thus, the DNN should be limited in the number of layers, neurons, and therefore weights. The UAV ui receiving the packet starts building the IDS function ρ. It

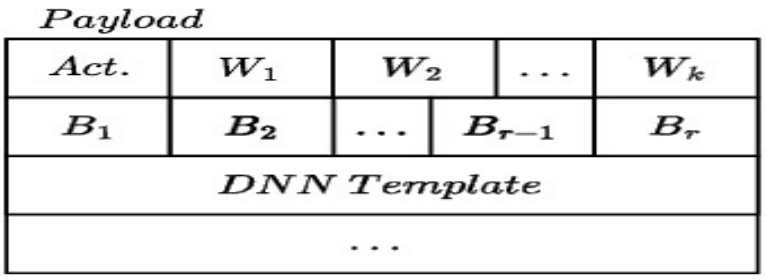

Fig. 2. Activation message: Custom payload of the TCP packet used to activate an IDS-enabled UAV.

extracts the weights $W1,W2,\ldots,Wk$, the bias values $B1,B2,\ldots,Br$, and the DNN template and uses both to set up ρui . Setting up ρui causes the UAV to experience a downtime period $Tdwn$, which can be defined as the time interval between the arrival of the activation TCP packet andthe configuration of ρ using the parameters extracted from that packet. After $Tdwn$, the IDS- enabled UAV ui is activated starting to overwatch the network classifying the network traffic.

4.2 Initial IDS Activation and Initialization Phase

The initial activation and initialization phase of an Intrusion Detection System (IDS) for a Flying Ad Hoc Network (FANET) is crucial for setting up a robust and functional security framework. This phase involves several steps to ensure that the IDS is properly configured, integrated, and ready to detect and respond to potential threats effectively. Here's a detailed breakdown of the activities and considerations for this phase:

4.2.1. System Preparation

- **Hardware Setup**: Ensure that all UAVs and other network components are physically prepared and operational. This includes verifying that sensors, communication modules, and computational resources are functioning correctly.
- **Software Deployment**: Install the IDS software on the designated UAVs and any central or distributed control systems. Ensure that the software versions are compatible and up-to-date.

4.2.2. Configuration and Calibration

- **Network Configuration**: Set up the network parameters, including IP addresses, communication protocols, and network topology. Ensure that all UAVs can communicate effectively with each other and with the control center.

- **IDS Parameters**: Configure IDS-specific parameters, such as detection thresholds,monitoring intervals, and alert settings. Tailor these settings to the specific requirements of the FANET and the expected types of threats.
- **Resource Allocation**: Allocate resources for IDS operations, including processing power, memory, and energy. Ensure that the resource allocation is balanced to prevent overloading individual UAVs.

4.3 Failure/load-balancing driven IDS-enabled UAV activation

After the activation of the first IDS-enabled drone during the Initial Activation Phase, it could be needed to activate more IDS-enabled UAVs in order to properly protect the fleet of UAVs that is providing a crucial service guaranteeing network connectivity. Considering a critical disaster event, this can be paramount taking into consideration two possible scenarios:

- Increased network traffic load.
- Failure of active IDS-enabled UAVs.

In the first scenario, the activation of a new IDS-enabled UAV is crucial in order to prevent the already deployed and activated IDS-enabled UAVs from being overloaded and potentially torn down. The failure of a UAV could be caused, in addition to the mentioned scenario, by other issues. For instance, the energy supply of the UAVs can be quickly and suddenly drained by the huge amount of traffic to be handled and classified. Or simply, a malfunction could occur tearingdown the vehicle. Due to these considerations, it is needed to define a procedure for the SDN controller to activate an IDS-enabled UAV whenever is needed or whenever a specific event is verified. When a new IDS-enabled UAV has to be activated due to the mentioned critical scenarios, the SDN controller executes the following steps

$$j \xleftarrow{R} [1, m] \ \wedge \ u_j \notin U_{IDS}^A$$
$$pick\ u_j \in U_{IDS}^I$$
$$U_{IDS}^A = U_{IDS}^A \bigcup \{u_j\}$$
$$U_{IDS}^I = U_{IDS} \setminus U_{IDS}^A \quad (14)$$

Once uj is randomly selected in the set of inactive IDS-enabled UAVs, the SDN controller and the activated UAV uj follow the same operation performed in the described Initialization Phase.

Failure/Load-Balancing Driven IDS-Enabled UAV Activation

In a Flying Ad Hoc Network (FANET) environment, managing the activation of IDS-enabled UAVs (Unmanned Aerial Vehicles) in response to failure or load-balancing needs is critical for maintaining network performance and security. This process involves dynamically adjusting the network to ensure that IDS tasks are distributed effectively and that UAVs are activated or reconfigured as needed. Here's a detailed breakdown of this process:

4.3.1 Failure Detection and Response

- **Health Monitoring**: Continuously monitor the status of UAVs, including battery levels, 2 processing health, communication link status, and sensor functionality.
- **Failure Identification**: Use predefined criteria and thresholds to detect failures or degraded performance in UAVs. This could include loss of communication, energy depletion, hardware malfunctions, or software errors.
- **Failure Isolation**: Isolate the failed or degraded UAV to prevent it from affecting the overall network performance. Disable or suspend its IDS tasks until it is repaired or replaced.
- **Task Reallocation**: Reassign IDS tasks from the failed UAV to other operational UAVs. Prioritize tasks based on the remaining capabilities and energy levels of available UAVs.

4.3.2 Load-Balancing Strategy

- **Resource Utilization Tracking**: Monitor resource usage on each UAV, including CPU, memory, network bandwidth, and energy consumption.
- **Load Metrics**: Collect metrics related to the current load on each UAV, such as the number of packets processed, number of alerts generated, and data throughput.
- **Dynamic Load Balancing**: Based on real-time monitoring, dynamically adjust the

- distribution of IDS tasks across UAVs to balance the load. This ensures that no single UAV is overloaded and that resources are utilized efficiently.
- **Task Assignment Algorithms**: Implement algorithms for task assignment that consider UAVs' current load, resource availability, and operational status. Common algorithms include round-robin, least-loaded, and priority-based scheduling.
- **UAV Activation**: Activate additional UAVs as needed to handle increased load or to replace UAVs that are unable to perform their IDS functions. This may involve deploying standby UAVs or reallocating tasks from less critical to more capable UAVs.
- **UAV Deactivation**: Deactivate or reduce the IDS tasks of UAVs that are operating below optimal performance levels or are nearing energy depletion. Transition their tasks to more capable UAVs to maintain system stability.

4.3.3 Coordination and Communication

- **Centralized Coordination**: In cases where a central control unit is present, use it to coordinate task redistribution, monitor network health, and manage UAV activation and deactivation.
- **Distributed Coordination**: In decentralized or peer-to-peer configurations, implement protocols that allow UAVs to negotiate task redistribution and manage load balancing autonomously.
- **Real-Time Updates**: Ensure that communication protocols support real-time updates and synchronization of task assignments and status information across UAVs.
- **Failover Communication**: Implement communication failover mechanisms to ensure that coordination messages are delivered even if some UAVs fail or experience connectivity issues.

4.3.4 System Optimization and Maintenance

- **Algorithm Tuning**: Regularly review and adjust load-balancing algorithms and failure response strategies based on system performance data and feedback.
- **Resource Management**: Optimize resource allocation to balance processing power, memory, and energy consumption across the network.
- **System Health Checks**: Perform regular maintenance checks on UAVs to ensure they are

- operating correctly and to prevent potential failures.
- **Software Updates**: Keep IDS and load-balancing software up-to-date to incorporate the latest improvements and security patches.
- **Continuous Monitoring**: Establish a feedback loop that continuously monitors system performance and gathers data on failures, load distribution, and task management.
- **Adaptation and Improvement**: Use feedback data to refine failure detection mechanisms, load-balancing algorithms, and overall system configuration to enhance reliability and performance.

4.3.5 Documentation and Reporting

- **Configuration Records**: Maintain records of IDS configurations, load-balancing settings, and failure response procedures.
- **Incident Logs**: Document incidents related to UAV failures, load imbalances, and task reallocations to aid in troubleshooting and system improvement.
- **Performance Reports**: Generate regular reports on system performance, including load distribution, task completion, and failure rates.
- **Incident Reports**: Provide detailed reports on failures, including causes, responses, and outcomes, to support post-incident analysis and continuous improvement.

By following these guidelines, you can effectively manage the activation and deactivation of IDS-enabled UAVs in response to failures and load-balancing needs. This approach helps ensure that the FANET remains secure, balanced, and operational even in the face of network challenges.

4.4 IDS function updating: Adaptive network awareness

Each UAV $ui \in UIDS$ is equipped with a knowledge base $\rightarrowtail ui$ which is used to make the UAV capable of being aware of the current network traffic condition, including all the threats that it has detected as well as the benign traffic that it has let to pass through it. However, the network is a highly dynamic entity meaning that the traffic conditions may change over time showing not only different traffic patterns but also new types of anomalies that should be carefully analyzed.

DNN performance evaluation.

Precision	Recall	F1-Score	Accuracy
0.994	0.995	0.994	0.998
False negative rate		False positive rate	
0.0023		0.0019	
True negative rate		True true rate	
0.9977		0.9981	

Fig 4.3 DNA Performance Evaluation

This dynamic nature of the network must be also captured by the IDS enabled UAVs to correctly adapt their traffic classification analysis to the utmost updated network conditions. To achieve this objective, while the IDS-enabled UAV performs its classification task on a network flow $f \in F$, it stores the network features extracted on that flow Xf along with the classification outcomes $d \in D$ as follows:

$$\forall f \in F$$
$$\sigma(f) = X_f \in \Gamma$$
$$\rho(X_f) = d \in D$$
$$add \{X_f, d\} to \Xi_{u_i}$$

The updated knowledge base is then used to update the IDS func tion ρui (for each IDS-enabled UAV) in order to achieve the Adaptive Network Awareness (ANA) feature:

$\rho ui [\rightarrowtail ui] = \rho' ui$

In such a way, the IDS function will be constantly adapted to the current network traffic profile as well as to the new anomalies that could arise with the dynamic change of the network and the new threats affecting the network. This update is performed as a background process to avoid blocking or affecting the IDS function of the UAV. After the update has been completed the knowledge base $\rightarrowtail ui$ is zeroed out to avoid letting it arbitrarily increase.

4.5 IDS function ρ implementation: DNN design

The ρ IDS function has been implemented using a DNN. More specifically, the considered DNN has been designed considering a trade-off between the model complexity and its accuracy in discerning benign traffic and anomalies. In Fig. 3 the designed DNN is shown. Itis composed of an input layer with 72 neurons followed by three hidden layers with 12, 6, and 3 neurons respectively. All of the hidden layers adopt the ReLu activation function. The output layer, which uses the Softmax activation function, is composed of two neurons since two are the possible labels for the considered classification task: 0- Benign and 1- Malicious. The DNN has been trained using the widely adopted dataset, which is composed of real network traffic samples. A training test split ratio of 70%–30% has been considered. The training procedure has been assisted using the Early Stopping technique, which ended the training after 19 epochs. The Adam optimizer with a learning rate lr = 0.01 has been employed, considering a batch size of 100 samples. In equation (15)

$$FPr = \frac{FP}{FP + TN}$$
$$FNr = \frac{FN}{FN + TP}$$

The performance of the obtained DNN is summarized instead, highlights the performance of the models in terms of True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN) by means of the model's Confusion Matrix. The model showed an accuracy of ~99% along with high values of recall, precision, and F1-score. However, these metrics are not enough to evaluate the model. Indeed, the False Positive and False Negative rates(FPr and FNr respectively) are paramount when dealing with intrusion detection. To compute them, the following formulas can be used:

By using the confusion matrix shown in Fig. 4(a) these values have been computed. The results prove that both FPr and FNr are negligible (0.0019 and 0.0023 respectively) meaning that the model shows a high confidence in distinguishing benign and malicious samples. Therefore, these results prove that the model complexity is not traded with its performance in detecting malicious events. As an additional proof of the accuracy obtained by the considered DNN, in Fig. 4(b) the results of a validation phase are given. More specifically, during training, 20% of the training dataset has been used to validate the DNN over each epoch. The training and validation accuracy showed a similar and almost over lapped trend.

This means that the DNN is able to perform with the same accuracy (which is ~99%) not only on the training set but also on a set of unseen data, i.e. the validation set, showing a good

generalization degree. The same trend can be observed for the loss curves in Fig. 4(c) further confirming the results described so far. These results prove that the obtained DNN does not trade a lesser complexity as well as a lightweight nature with the overall performance in detecting malicious network flows among the normal ones.

Finally, the resulting DNN has been encoded into a DNN template which describes the structure of the network in terms of, layers, neurons per layer, and activation functions using a JSON representation. Also, the weights and the bias values have been included in the TCP Activation Message Packet, as described in Fig. 2. The lesser complexity of the DNN also affects the network footprint required to transmit the JSON representations through the TCP packet.

More specifically, the DNN has a total number of trainable parameters equal to 983 (960 weights and 23 biases). Considering a 4-byte floating-point representation for each value, the total amount of bytes required to transmit all the parameters is 3932 bytes. Concerning the DNN template, instead, the JSON representation requires 3178 bytes. Therefore, the network footprint required to transmit the DNN template along with the trainable parameters is equal to 7110 bytes.This footprint can be quantified as ∼ 11% of the maximum allowed size for a single TCP packet. Once more, the simplified structure of the DNN under consideration yields significant advantages in network footprint, reducing it to a negligible size. Finally, the σ function has been implemented to extract from each network flow the related features that are used to feed the DNN to classify that flow.

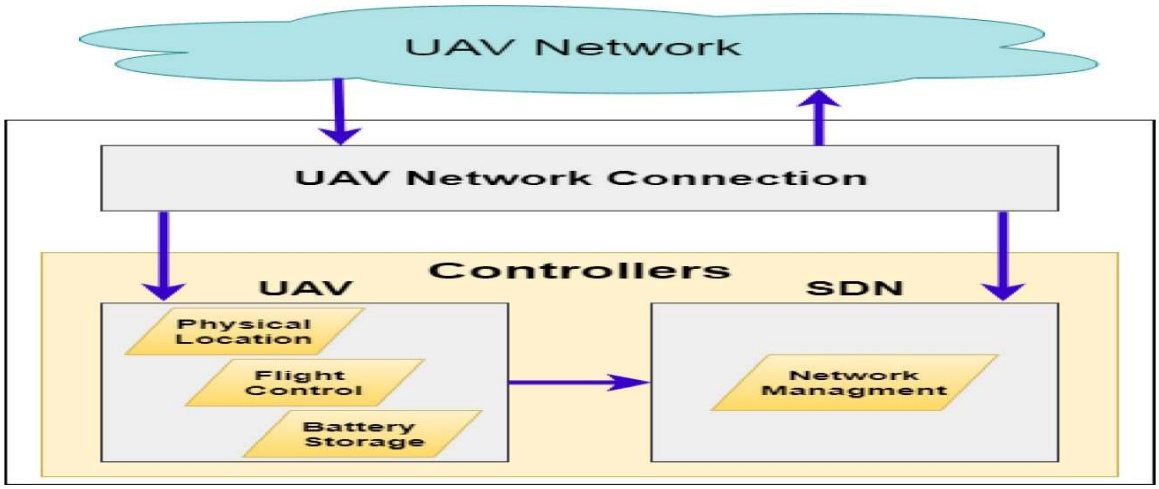

Fig.4. Block diagram of UAV network architecture

In the SDN based UAV network, UAVs implement SDN features and the ground station such as the SDN controller as well as user equipment (UE) for control failures. In the SDN-based UAV network, parameters of UAVs and network statistics are collected by the SDN controller. After that, the final optimal decision is taken by utilizing the precise computed results.

The functional architecture of the UAV network is shown in Fig.4 with SDN controllers for the UAV network. The UAV controller manages information such as physical location, battery storage, and flight control and the SDN controller is to interact with the UAV controller and distribute the information about UAVs network. When the SDN controller detects a poor wireless link state, a message is sent to the UAV controller by the SDN controller. Then, the UAV adjusts the position with the intention of having an improved and stable communication link according to the command of the UAV controller. Significant strategies can be made based on the analysis and management of these statistics, which are from the network and UAVs. As mentioned earlier, managing the vast bulk of data is a problem.

Summary:

System analysis for a software-defined IDS in FANET disaster scenarios focuses on the initial activation and initialization of IDS nodes, ensuring real-time detection and response. It involves monitoring for UAV failures and dynamically reassigning tasks based on load andresource availability. Load-balancing strategies must consider UAV capabilities and energy constraints to optimize performance. Coordination among UAVs and the control center is essential for seamless operation and task redistribution. Continuous feedback and adaptation are crucial for maintaining system effectiveness and reliability in dynamic disaster environments.

CHAPTER 5

SYSTEM DESIGN

5.1 Experimental setup

In order to evaluate the proposal, the experimental SDN-FANET test bed has been built using the Mini net-WiFi network emulator. The UAVs have been modeled and implemented as Ubuntu Docker containers. The Ryu SDN controller has been chosen. Each UAV has been integrated into the SDN-FANET by deploying and executing an Open vSwitch (OvS) 2.13.8 [26] within it. The UAVs of the networks are connected through wireless links, using the WiFi 802.11 technology, forming the data plane of the support network, as shown in Fig. 1. The SDN controller manages the data plane UAVs using the $SBI1$ which has been implemented by means of the Open Flow v1.3 protocol. This implementation facilitates the addition, modification, and removal of flow rules, ensuring effective and flexible management of network flows traversing the FANET. The wireless channel, which is crucial when considering an Ad-Hoc network like FANET, poses a great challenge concerning transmission technology standards when dealing with OpenFlow-based networks. More specifically, Open Flow is primarily designed for use in wired networks, particularly on Ethernet 802.3-based networks. However, a wired transmission means is not appropriate for FANET due to its inherent nature. To overcome this issue, and be able to adapt OpenFlow to a wireless FANET scenario it has been resorted to VXLAN technology to create direct tunnels among the UAVs deployed in the SDN-FANET. Through these direct tunnels the,

Table 3
Simulation parameters.

Parameter	Value
$\|U\|$	13
$\|U_{IDS}\|$	2 to 5
Users	60
Wi-Fi technology	IEEE 802.11g
Maximum bit rate (B_{MAX})	54 Mbps
Frequency	2.4 GHz
Bandwidth	20 MHz
Modulation	OFDM
Propagation model	LogDistance
Simulation time	23 min

Table 3: Simulation Parameters

OpenFlow packets, encapsulated into UDP packets, can be transmitted by exploiting the WiFi 802.11. The characteristics of the wireless technology considered during the simulation are depicted in Table 3. All the experiments have been carried out on an Ubuntu Server equipped with an Intel(R) Xeon(R) Gold 6252N CPU @ 2.30 GHz with 192 CPU cores and 528 GB of memory. To quantify the energy consumption of the supplementary functions (φ and δ), referred to as $EFUN$, on each UAV, we employed PowerAPI, a widely utilized software-based power meter designed for virtualized systems. The topology depicted in Fig. 1 has been considered for the test campaigns. It has been considered a set U of UAVs with $|U|$ = 13. Among them, a set of IDS-enabled UAVs $UIDS$ such that $|UIDS|$ = 5. The UAVs are modeled as DJI MATRICE 100 [18] with 8 GB of RAM and 4 CPU cores. To evaluate the proposed load-balancing strategy compared it with baseline load-balancing strategies such as random and round robin [7] considering evaluation metrics such as CPU utilization, battery autonomy, and packet loss rate foreach IDS-enabled UAV. CPU/RAM utilization provided useful information about the computational workload imposed on each UAV.

The battery duration is paramount in UAV-based deployments and it is crucial to preserve it as much as possible to avoid UAVs being abruptly shut down. Finally, packet loss rate has been monitored to assess the reliability and integrity of data transmission within the network. By analyzing these performance metrics, we can gain a comprehensive understanding of the capabilities and limitations of different load-balancing techniques within the context of an SDN FANET enhanced with IDS-enabled UAVs. Additionally, we measured the classification time of the IDS function ρ, a critical metric indicative of system responsiveness and efficiency.

This classification time is made up of two fundamental components: the feature extraction time and the inference time required by the model to compute the outcome (i.e. malicious or not). The feature extraction time is the duration taken by the σ function to extract relevant features from the incoming network flows, and the inference time is the time taken for the IDS function to process the extracted features and generate a classification.

5.2 Simulation scenarios

The experiments have been carried out considering four cases:

i. without load-balancing where there exists just one IDS-enabled UAV within the network,

ii. random load-balancing,

iii. round-robin load balancing, and

iv. The proposed load-balancing.

In the scenarios with load-balancing strategies the subset $Uids$ dimension variates from 2 to 5 IDS-enabled drones. To evaluate the impact of the IDS module, the re sults were compared with a baseline scenario (referred to with a dashed line in the figures) in which UAVs only act as forwarding entities (i.e. execute only the φ function, see Section 3.1). In the first simulation scenario, we assessed the performance of the proposed load-balancing strategy when no attackers were misbehaving within the SDN-FANET. For such a scenario, we considered a set of 60 users that dynamically join the SDN-FANET generating TCP traffic.

Each user joins the network every five seconds, starting transmitting the TCP traffic (approximately 30 pkts/s) towards a target user, for the whole simulation which lasts for 300 s. The other considered scenario takes into consideration a situation in which the SDN-FANET is subjected to intrusion by malicious entities (attackers). More specifically, we considered ascenario in which five IDS-enabled UAVs are available (i.e. $|UIDS| = 5$) and with the number of attackers growing from two to ten with an increasing step of two until reaching a total of 10 simultaneous attackers. Each attacker generates malicious traffic at a rate of 100 pkts/s with a random payload whose size is $\sim$1250 bytes, targeting randomly chosen IP addresses among the set of targets. The considered attack that is perpetrated by each of the malicious entities is a SYNFlooding Distributed Denial of Service (DDoS). This last test is aimed at replicating a dynamic and evolving threat scenario, allowing for the comprehensive evaluation of the resilience of the network and the effectiveness of the IDS.

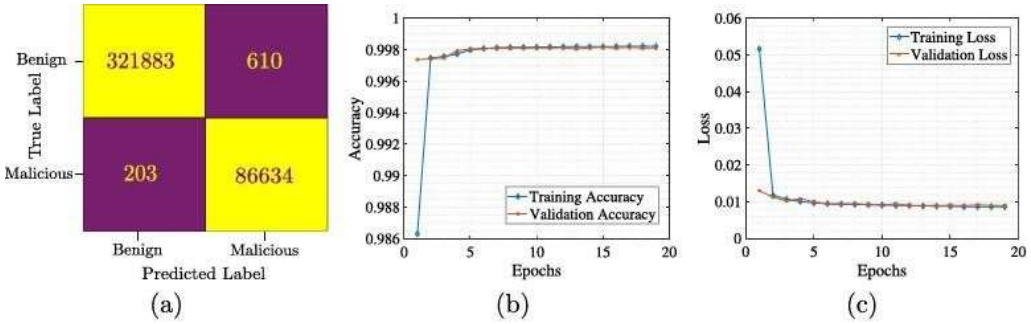

Fig. 4. (a) DNN's confusion matrix. (b) Training vs. validation accuracy. (c) Training vs. validation loss.

By using the confusion matrix shown in Fig. 4(a) these values have been computed. The results prove that both FPr and FNr are negligible (0.0019 and 0.0023 respectively) meaning that the model shows a high confidence in distinguishing benign and malicious samples. Therefore, these results prove that the model complexity is not traded with its performance in detecting malicious events. As an additional proof of the accuracy obtained by the considered DNN, in Fig. 4(b) the results of a validation phase are given. More specifically, during training, 20% ofthe training dataset has been used to validate the DNN over each epoch. The training and validation accuracy showed a similar and almost over lapped trend. This means that the DNN is able to perform with the same accuracy (which is ~99%) not only on the training set but also ona set of unseen data, i.e. the validation set, showing a good generalization degree. The same trendcan be observed for the loss curves in Fig. 4(c) further confirming the results described so far. These results prove that the obtained DNN does not trade a lesser complexity as well as a lightweight nature with the overall performance in detecting malicious network flows among the normal ones. Finally, the resulting DNN has been encoded into a DNN template which describes the structure of the network in terms of, layers, neurons per layer, and activation functions usinga JSON representation.

5.3 Load-balancing coefficients analysis

To conduct an experimental analysis to determine the values of the coefficients in Eq. (26) that resulted in a balanced trade-off among all the mentioned performance metrics. Fig. 5illustrates the relationship between battery autonomy, measured in minutes, and CPU usage,

expressed in percentage (%). The simulation was conducted using different coefficients in the proposed load-balancing algorithm considering a scenario with 5 IDS-enabled nodes. Each point in the figure represents different combinations of coefficients. Notably, points with coefficients that place more weight on the battery level exhibit longer battery duration using less CPU. Conversely, points, where CPU usage carries more weight, tend to result in shorter battery auton omy. Upon analysis of the figure, it becomes evident that the most favorable outcomes are achieved when coefficients are homogeneously distributed across the four factors, suggesting that a balanced strategy gives optimal results.

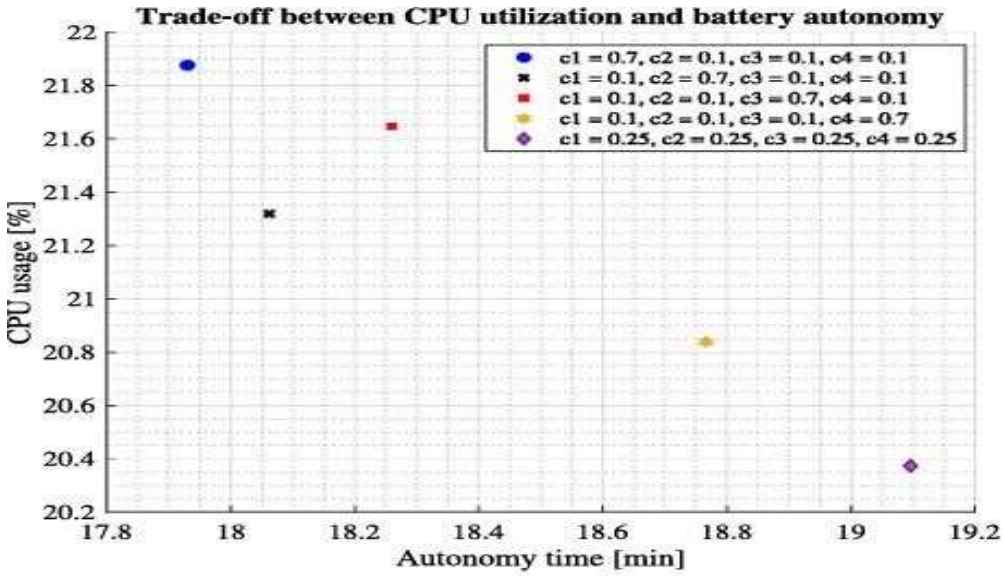

Fig. 5. Coefficients Analysis of the Eq. (26).

According to the results, the best combination of coefficients is$c1 = 0.25,2 = 0.25,3 = 0.25,c4 = 0.25$, which provides large battery duration of the IDS- enabled nodes consuming the less possible CPU without affecting the network performance. Therefore, the simulations with the proposed load-balancing module are carried out using the combination of these coefficients.

5.3.1 Load-Balancing Coefficients Analysis

Load-balancing coefficients are critical in optimizing the distribution of IDS (Intrusion Detection System) tasks across UAVs (Unmanned Aerial Vehicles) in a Flying Ad Hoc Network (FANET). These coefficients help in deciding how to allocate tasks based on various metrics and constraints, ensuring efficient resource utilization and maintaining network performance. Here's

are carried out using the combination of these coefficients.

5.3.2 Definition and Purpose

Coefficients Overview

- **Load Coefficient**: A measure representing the current load on a UAV, such as CPU usage, memory consumption, or data processing workload.
- **Resource Coefficient**: Indicates the availability of resources on a UAV, including processing power, memory, and battery life.
- **Priority Coefficient**: Reflects the importance or priority of specific tasks that need to be balanced, such as high-priority security checks or critical data analysis.
- **Fairness Coefficient**: Ensures that load distribution is fair across all UAVs, preventing any single UAV from being overburdened.

5.3.3 Purpose

- **Optimal Resource Utilization**: Ensure that no single UAV is overloaded, and resources are utilized efficiently across the network.
- **System Stability**: Maintain network stability and performance by balancing the workload according to the capabilities of each UAV.
- **Energy Efficiency**: Distribute tasks in a way that optimizes energy consumption and extends the operational time of each UAV.

5.3.4 Load-Balancing Metrics

- **CPU Utilization**: The percentage of CPU capacity currently in use by the IDS processes.
- **Memory Usage**: The amount of memory being utilized for IDS operations.
- **Network Bandwidth**: The amount of network bandwidth consumed by data transmission and reception.
- **Remaining Battery Life**: The estimated remaining operational time based on current energy consumption.
- **Processing Power**: The available computational power of the UAV, typically measured in processing cycles or speed.
- **Storage Capacity**: The available storage for logging and analyzing network traffic.

- **Task Criticality**: The urgency and importance of different IDS tasks, which might affect their priority in load balancing.
- **Alert Severity**: The severity level of alerts generated by the IDS, influencing task assignment to prioritize higher-severity issues.
- **Load Distribution**: Measures how evenly the workload is distributed among all UAVs.
- **Resource Allocation Equality**: Ensures that each UAV receives a proportionate share of the total resources.

5.3.5 Coefficient Calculation

Formulae for Coefficient Calculation

- **Load Coefficient (LC)**: LCi=LiRiLC_i = \frac{L_i}{R_i}LCi=RiLi,

.2 where LiL_iLi is the load on UAV iii, and RiR_iRi is the resource availability of UAV

- **Resource Coefficient (RC)**: RCi=RiTiRC_i = \frac{R_i}{T_i}RCi=TiRi,

.2 2.4.2.1 where TiT_iTi is the total resources required for the tasks assigned to UAV iii.

- **Priority Coefficient (PC)**: PCt=Pt∑iPiPC_t = \frac{P_t}{\sum_{i} P_i}PCt=∑iPiPt, where PtP_tPt is the priority of task ttt, and ∑iPi\sum_{i} P_i∑iPi is the total priority of all tasks.
- **Fairness Coefficient (FC)**:FC=max(LCi)−min(LCi)max(LCi)+min(LCi)FC =
- \frac{\text{max}(LC_i)\text{min}(LC_i)}{\text{max}(LC_i) +
- \text{min}(LC_i)}FC=max(LCi)+min(LCi)max(LCi)−min(LCi), measuring the balance of load distribution.
- **Weighted Load Coefficient**: Incorporate weights based on task importance or UAV capabilities. For example, LCi=w1·LiRi+w2·PiRiLC_i = w_1 \cdot \frac{L_i}{R_i} + w_2

\cdot \frac{P_i}{R_i}LCi=w1·RiLi+w2·RiPi, where w1w_1w1 and w2w_2w2 are weights assigned to load and priority.

5.3.6 Analysis and Optimization

- **Historical Data Review**: Analyze historical load-balancing data to understand patterns and adjust coefficients accordingly.
- **Performance Metrics**: Review system performance metrics such as task completion

times, alert accuracy, and resource utilization efficiency to validate the effectiveness of the coefficients.

- **Dynamic Adjustment**: Continuously adjust coefficients in real-time based on changing network conditions, UAV statuses, and load variations.
- **Algorithm Tuning**: Refine load-balancing algorithms by tuning coefficients to enhance performance and fairness. This might involve iterative testing and adjustment.
- **Simulation Models**: Use simulations to test different coefficient settings and their impact on load distribution, system stability, and resource utilization.
- **Scenario Testing**: Validate coefficient effectiveness under various scenarios, such as peak traffic loads, UAV failures, and changes in network topology.

5.3.7 Implementation Considerations

- **System Integration**: Integrate coefficient calculations and adjustments into the IDS management system.
- Ensure that the system can dynamically compute and apply coefficients as conditions change.
- **Real-Time Processing**: Ensure that the coefficient calculations and load-balancing decisions are made in real-time to respond promptly to network conditions.
- **Continuous Monitoring**: Continuously monitor coefficient performance and make adjustments based on real-time data and feedback.
- **User Feedback**: Gather feedback from system operators and analysts to understand the practical implications of coefficient settings and make necessary refinements.
- By carefully analyzing and optimizing load-balancing coefficients, you can ensure that IDS-enabled UAVs in a FANET are utilized effectively, balancing the load and maintaining optimal performance and security across the network.

5.4 Activation responsiveness and classification time evaluation

Finally, The evaluated the responsiveness of the activation procedure when a failure, of an IDS- enabled UAV, occurs. For such a reason to conduct an experiment with $UIDS = 2$. Initially, only one IDS-enabled UAVis active, such that $UA\ IDS = 1$. After 180 s, tore down the active IDS-enabled UAV letting the SDN controller enable the new one in the set $UI\ IDS$, as explained

in Section 4.2. In Fig. 13(a) to show the trend of the CPU utilization over time. While the first active IDS-enabled UAV is working, the inactive one is also serving as a forwarding entity (executing only the φ function). Once the failure occurs, around 180 s, the other IDS-enabled UAV must be activated by the SDN Controller to act as IDS.

While the first active IDS-enabled UAV is working, the inactive one is also serving as a forwarding entity (executing only the φ function). Once the failure occurs, around 180 s, the other IDS-enabled UAV must be activated by the SDN Controller to act as IDS. The two red circles represent the start and the end points of the initialization procedure showcasing thequicker activation of a new IDS-enabled UAV when a failure occurs. Indeed, less than 10 s are needed to let the new IDS-enabled UAV be active to classify network flows.

The two red circles represent the start and the end points of the initialization procedure showcasing the quicker activation of a new IDS-enabled UAV when a failure occurs. Indeed, less than 10 s are needed to let the new IDS-enabled UAV be active to classify network flows. This is a crucial result, especially when dealing with disaster events. Along with the activation responsiveness, The evaluated the IDS enabled UAVs' effectiveness in performing network flow classification when the workload is fairly distributed on them using the proposed-LB. This evaluation was conducted in comparison to the baseline strategies, i.e. Random and Round Robin. Thanks to the resource-aware strategy, and as previously demonstrated, each IDS-enabled UAV is not overwhelmed by network flows to be analyzed. This results in better resource management, which leads the IDS-enabled UAVs to produce the classification in a shorter time with respect to the other load balancing strategies, that are resource-agnostic.

Summary:

The system design for a software-defined IDS in FANET disaster scenarios incorporates a modular architecture with real-time anomaly detection, dynamic load-balancing, and resource management. It includes UAVs equipped with sensors and processing units for data collection and analysis, while coordinating tasks based on UAV resource availability and operational status. Load-balancing algorithms are employed to distribute IDS tasks efficiently, considering factors likeCPU usage, memory, and battery life.

CHAPTER 6

IMPLEMENTATION

The software-defined intrusion detection system with UAV resource-aware load balancing for FANETs in disaster scenarios:

Implementation Steps:

6.1 IDS Implementation:

- Develop IDS using machine learning algorithms and statistical analysis
- Integrate IDS with SDN Controller and URLB
- Deploy IDS on UAVs

Implementing a software-defined Intrusion Detection System (IDS) with UAV resource-aware load-balancing in a Flying Ad Hoc Network (FANET) for disaster scenarios involves several key steps to ensure efficient and effective operation.

6.2 URLB Implementation:

- Develop URLB using dynamic load balancing algorithms
- Integrate URLB with SDN Controller and IDS
- Deploy URLB on UAVs

The URLB (Unified Resource-Aware Load-Balancing) implementation for a software-defined IDS (Intrusion Detection System) in FANET (Flying Ad Hoc Network) disaster scenarios involves a comprehensive approach to manage UAV (Unmanned Aerial Vehicle) resources effectively while maintaining optimal network security

6.3 SDN Controller Implementation:

- Develop SDN Controller using centralized security configuration and policy enforcement
- Integrate SDN Controller with IDS and URLB
- Deploy SDN Controller on a central server

The implementation of an SDN (Software-Defined Networking) controller for managing a software-defined IDS (Intrusion Detection System) with UAV (Unmanned Aerial Vehicle) resource-aware load-balancing in a FANET (Flying Ad Hoc Network) disaster scenario.

6.4 UAV Resource Monitoring:

- Develop UAV resource monitoring system to track battery life, processing power, and communication bandwidth
- Integrate UAV resource monitoring with URLB

Effective UAV resource monitoring is crucial for the successful implementation of a software-defined IDS with UAV resource-aware load-balancing in a FANET (Flying Ad Hoc Network) disaster scenario. This component ensures that the IDS tasks are appropriately allocated based on the current state of each UAV, thereby maintaining network efficiency and security.

6.5 System Integration:

- Integrate IDS, URLB, and SDN Controller
- Test system for functionality and performance

System integration is a critical phase in the deployment of a software-defined IDS (Intrusion Detection System) with UAV (Unmanned Aerial Vehicle) resource-aware load-balancing in a FANET (Flying Ad Hoc Network) disaster scenario. It involves combining various system components to work seamlessly together, ensuring optimal performance and functionality

6.6 Deployment:

- Deploy system in a disaster scenario

- Monitor system performance and adjust as needed.

The deployment of a software-defined IDS (Intrusion Detection System) with UAV (Unmanned Aerial Vehicle) resource-aware load-balancing in a FANET (Flying Ad Hoc Network) disaster scenario involves several critical steps to ensure that the system operates effectively in real-world conditions.

6.7 CODE:

```
import matplotlib.pyplot as plt
import numpy as np
import random



# UAV structure
class UAV:
    def _init_(self, uav_id, x, y, z, energy):
        self.uav_id = uav_id
        self.x = x
        self.y = y
        self.z = z
        self.energy     =     energy
self.is_intruder = False uavs = []

def main():

    sdn_controller = SDNController()
    initialize_uavs(sdn_controller, 10)


    print("Initial UAVs:")
    display_uavs()


    sdn_controller.detect_intruders()
    print("After Intruder Detection:")
    display_uavs()
```

```
# SDN Controller to manage FANETs
 class SDNController:
    def _init_(self):
        self.uavs = []
 def register_uav(self, uav):
        self.uavs.append(uav)
 def detect_intruders(self, threshold=30.0):
        for uav in self.uavs:
            if uav.energy < threshold:
                uav.is_intruder = True


 def load_balance_tasks(self):

        tasks = [random.uniform(5, 20)for _ in range(len(self.uavs))]
for i, uav in enumerate(self
self.is_intruder = False uavs = []


# SDN Controller to manage UAVs class
SDNController:
     def _init_(self):
        self.uavs = []
  def register_uav(self, uav):
        self.uavs.append(uav)


 class SDNController:
    def _init_(self):
        self.uavs = []
 def register_uav(self, uav):
        self.uavs.append(uav)
 def detect_intruders(self, threshold=30.0):
        for uav in self.uavs:
            if uav.energy < threshold:
                uav.is_intruder = True
 def load_balance_tasks(self):

 def detect_intruders(self, threshold=30.0):
        for uav in self.uavs:
            if uav.energy < threshold:
                uav.is_intruder = True
```

self.is_intruder = False uavs = []

6.1.2 Implementing a Software-Defined IDS with UAV Resource-Aware Load Balancing in FANETs for Disaster Response

```
 # SDN Controller to manage UAVs
 class SDNController:
     def _init_(self):
         self.uavs = []
  def register_uav(self, uav):
         self.uavs.append(uav)
  def detect_intruders(self, threshold=30.0):
         for uav in self.uavs:
             if uav.energy < threshold:
                 uav.is_intruder = True
  def load_balance_tasks(self):

         tasks = [random.uniform(5, 20) for _ in range(len(self.uavs))]
         for i, uav in enumerate(self.uavs):
             if not uav.is_intruder:
                 uav.energy -= tasks[i]
                 if uav.energy < 0:
                     uav.energy = 0

  def detect_intruders(self, threshold=30.0):
         for uav in self.uavs:
             if uav.energy < threshold:
                 uav.is_intruder = True


  def load_balance_tasks(self):

         tasks = [random.uniform(5, 20)for _ in range(len(self.uavs))]
for i, uav in enumerate(self
self.is_intruder = False uavs = []


# SDN Controller to manage UAVs class
SDNController:
     def _init_(self):
         self.uavs = []
  def register_uav(self, uav):
         self.uavs.append(uav)
```

```
    def manage_fanet(self):
        for uav in self.uavs:
            if not uav.is_intruder:

                uav.x += random.uniform(-1, 1)
                uav.y += random.uniform(-1, 1)
                uav.z += random.uniform(-1, 1)
 def simulate_disaster(self):
        for uav in self.uavs:
            if random.random() < 0.1: # 10% chance of failure
                uav.is_intruder = True
def initialize_uavs(controller, num_uavs):
    global uavs
    uavs = []

    for i in range(num_uavs):

        uav = UAV(i, random.uniform(0, 100), random.uniform(0, 100), random.uniform(0, 100),
100.0)
        uavs.append(uav)
        controller.register_uav(uav)

def display_uavs():

    fig = plt.figure()

    ax = fig.add_subplot(111, projection='3d')
    for uav in uavs:
        color = 'r' if uav.is_intruder else 'g'
        ax.scatter(uav.x, uav.y, uav.z, c=color)
        ax.text(uav.x, uav.y, uav.z, '%d' % uav.uav_id, size=10, zorder=1, color='k')
    plt.show()


def main():

    sdn_controller = SDNController()
    initialize_uavs(sdn_controller, 10)


    print("Initial UAVs:")
    display_uavs()
```

```
    fig = plt.figure()

    ax = fig.add_subplot(111, projection='3d')
    for uav in uavs:
        color = 'r' if uav.is_intruder else 'g'
        ax.scatter(uav.x, uav.y, uav.z, c=color)
        ax.text(uav.x, uav.y, uav.z, '%d' % uav.uav_id, size=10, zorder=1, color='k')
    plt.show()


def main():

    sdn_controller = SDNController()
    initialize_uavs(sdn_controller, 10)


    print("Initial UAVs:")
    display_uavs()


    sdn_controller.detect_intruders()
    print("After Intruder Detection:")
    display_uavs()


    sdn_controller.load_balance_tasks()
    print("After Load Balancing:")

    def manage_fanet(self):
        for uav in self.uavs:
            if not uav.is_intruder:

                uav.x += random.uniform(-1, 1)
                uav.y += random.uniform(-1, 1)
                uav.z += random.uniform(-1, 1)
 def simulate_disaster(self):
        for uav in self.uavs:
            if random.random() < 0.1: # 10% chance of failure
                uav.is_intruder = True
```

```
    display_uavs()

    sdn_controller.manage_fanet()
    print("After FANET Management:")
    display_uavs()


    sdn_controller.simulate_disaster()
    print("After Disaster Simulation:")
    display_uavs()


if _name_ == "_main_":
    main()


import matplotlib.pyplot
    as plt
import numpy as np



# Data from the figures
    provided
```

6.1 packet loss in disaster

```
cpu_without_lb = [80, 60,
cpu_sura_lb = [40, 35, 30,

    25, 20]

cpu_no_ids = [0, 0, 0, 0,

    0]

ids_labels = ['1 IDS-
    enabled', '2 IDS-
    enabled', '3 IDS-
    enabled', '4 IDS-
    enabled', '5 IDS-
    enabled']
coeff_var_cpu = [31.2,
```

```
    50, 45, 40]

cpu_random_lb = [60, 50,

    40, 35, 30]

cpu_round_robin_lb =
    [50, 40, 35, 30, 25]
cpu_sura_lb = [40, 35, 30,

    25, 20]

cpu_no_ids = [0, 0, 0, 0,

    0]

ids_labels = ['1 IDS-
    enabled', '2 IDS-
    enabled', '3 IDS-
    enabled', '4 IDS-
    enabled', '5 IDS-
    enabled']
coeff_var_cpu = [31.2,
    32.2, 5.6]
lb_methods_cpu =
    ['Random LB', 'Round
    Robin LB', 'S_URA
    LB']
    display_uavs()

    sdn_controller.manage_fanet()
    print("After FANET Management:")
    display_uavs()


    sdn_controller.simulate_disaster()
    print("After Disaster Simulation:")
    display_uavs()
```

```
battery_without_lb = [10,
12, 14, 16, 18]
 battery_random_lb = [12,
    14, 16, 18, 20]
 battery_round_robin_lb =
    [14, 16, 18, 20, 22]
 battery_sura_lb = [16, 18,
    20, 22, 24]
 battery_no_ids = [25, 25,
    25, 25, 25]
 coeff_var_battery = [16.8,
    14.7, 3.7]

 lb_methods_battery =
    ['Random LB', 'Round
    Robin LB', 'S_URA
    LB']

fig, axs = plt.subplots(2, 2, figsize=(14, 10))
# packet loss co-efficient
 axs[0, 0].plot(ids_labels,
    cpu_without_lb,
    label='Without LB',
    color='blue')


 axs[0, 0].plot(ids_labels,
    cpu_random_lb,
    label='Random LB',
    color='orange')
 axs[0, 0].plot(ids_labels,
   cpu_round_robin_lb,
   label='Round  Robin
    LB', color='purple')
 axs[0, 0].plot(ids_labels,
    cpu_sura_lb,
    label='S_URA LB',
    color='red')
 axs[0, 0].plot(ids_labels,
    cpu_no_ids, label='No
    IDS function',
    linestyle='--',
    color='black')
```

```
 axs[0, 0].plot(ids_labels,
    cpu_random_lb,
    label='Random LB',
    color='orange')
 axs[0, 0].plot(ids_labels,
   cpu_round_robin_lb,
   label='Round  Robin
    LB', color='purple')
 axs[0, 0].plot(ids_labels,
    cpu_sura_lb,
    label='S_URA LB',
    color='red')
 axs[0, 0].plot(ids_labels,
    cpu_no_ids, label='No
    IDS function',
    linestyle='--',
    color='black')
 axs[0, 0].set_xlabel('IDS-
    enabled')
 axs[0, 0].set_ylabel('CPU
    [%]')
battery_without_lb = [10,
12, 14, 16, 18]
 battery_random_lb = [12,
    14, 16, 18, 20]
 battery_round_robin_lb =
    [14, 16, 18, 20, 22]
 battery_sura_lb = [16, 18,
    20, 22, 24]
 battery_no_ids = [25, 25,
    25, 25, 25]
 coeff_var_battery = [16.8,
    14.7, 3.7]

 lb_methods_battery =
    ['Random LB', 'Round
    Robin LB', 'S_URA
    LB']

fig, axs = plt.subplots(2, 2, figsize=(14, 10))
# packet loss co-efficient
 axs[0, 0].plot(ids_labels,
    cpu_without_lb,
    label='Without LB',
    color='blue')
```

```
axs[0,

    0].set_title('Average
    CPU')
axs[0, 0].legend()



#CPU co-efficient
axs[0,
    1].bar(lb_methods_cpu

    , coeff_var_cpu,
    color='blue')
axs[0,

    1].set_ylabel('Coefficie
    nt variation [%]')
axs[0, 1].set_title('CPU
    coefficient of
    variation')
# Average autonomy time
axs[1, 0].plot(ids_labels,
   battery_without_lb,
   label='Without LB',
    color='blue')
axs[1, 0].plot(ids_labels,
    battery_random_lb,
    label='Random LB',
    color='orange')
axs[1, 0].plot(ids_labels,
    battery_round_robin_lb
    , label='Round Robin
    LB', color='purple')
    axs[1, 0].plot(ids_labels,
```

```
 axs[1, 0].plot(ids_labels,
    battery_random_lb,
    label='Random LB',
    color='orange')
 axs[1, 0].plot(ids_labels,
    battery_round_robin_lb
    , label='Round Robin
    LB', color='purple')
 axs[1, 0].plot(ids_labels,
    battery_sura_lb,
    label='S_URA LB',
    color='red')
 axs[1, 0].plot(ids_labels,
    battery_no_ids,
    label='No IDS
    function', linestyle='--',
    color='black')
 axs[1, 0].set_xlabel('IDS-
    enabled')
 axs[1, 0].set_ylabel('Time
    [min]')

 axs[1,

    0].set_title('Average
    autonomy time')
 axs[1, 0].legend()
# autonomy time
axs[1,
    1].bar(lb_methods_batt
    ery,  coeff_var_battery,
    color='blue')
 axs[1,

    1].set_ylabel('Coefficie
    nt variation [%]')
 axs[1,

    1].set_title('Autonomy
    time coefficient of
    variation')


 plt.tight_layout()
plt.show()
```

Functions used:

- This code simulates **init_(self, uav_id, x, y, z, energy)**: Initializes a UAV instancewith ID, position coordinates, and energy level.
- **register_uav(self, uav)**: Adds a UAV to the SDNController's list of managedUAVs.
- **detect_intruders(self, threshold=30.0)**: Marks UAVs as intruders if their energyfalls below the specified threshold.
- **load_balance_tasks(self)**: Distributes tasks among UAVs and reduces their energybased on the assigned task load.
- **manage_fanet(self)**: Updates UAV positions randomly to simulate movement in theFANET.
- **simulate_disaster(self)**: Randomly marks some UAVs as intruders to simulatedisaster conditions.
- **initialize_uavs(controller, num_uavs)**: Creates and registers a specified number ofUAVs with the given SDNController.
- **display_uavs()**: Visualizes the UAVs' positions and statuses in a 3D scatter plot.
- **main()**: Runs the main simulation, including initialization, intruder detection, load balancing, FANET management, and disaster simulation.
- **plt.subplots()**: Creates a grid of subplots for plotting data.
- **axs[].plot()**: Plots lines on the subplots to visualize changes in CPU usage andbattery life.
- **axs[].bar()**: Creates bar charts to show variations in CPU and battery metrics.
- **plt.tight_layout()**: Adjusts subplot parameters to ensure that plots are neatlyarranged without overlapping.
- a scenario where a Software Defined Networking (SDN) controller manages a fleet of Unmanned Aerial Vehicles (UAVs) in a Flying Ad-Hoc Network (FANET). Here's a breakdown of the code:

UAV Class

- Represents an individual UAV with attributes:

- uav_id: unique identifier

is_intruder: boolean indicating if the UAV is an intruder

SDNController Class

- Manages the fleet of UAVs with methods:

- register_uav: adds a UAV to the controller's list

- detect_intruders: identifies UAVs with energy below a threshold as intruders

- load_balance_tasks: assigns tasks to UAVs, reducing their energy levels

- manage_fanet: updates UAV positions randomly

- simulate_disaster: simulates a disaster scenario where 10% of UAVs fail

Initialization and Simulation

- initialize_uavs: creates a specified number of UAVs and registers them with the SDN controller

- display_uavs: visualizes the UAVs in 3D using Matplotlib

- main: runs the simulation, calling the above methods in sequence

- The simulation demonstrates the following scenarios:

- Initial UAV deployment
- Intruder detection
- Load balancing and task assignment
- FANET management (UAV movement)
- Disaster simulation (UAV failure)

Summary:

The implementation of a software-defined IDS with UAV resource-aware load-balancing in FANET disaster scenarios involves deploying IDS software across UAVs, configuring real-time anomaly detection, and integrating load-balancing algorithms. Initial setup includes calibrating system parameters and establishing network communication for data exchange and coordination. Load-balancing mechanisms are activated to dynamically distribute IDS tasks based on UAV resource availability and current load. Continuous monitoring and adjustment ensure optimal performance and resource utilization. The system is tested through simulations and real-world scenarios to validate effectiveness and reliability in disaster conditions.

CHAPTER 7

TESTING

Testing the Software-Defined IDS with UAV Resource-Aware Load Balancing in FANET disaster scenarios involves a comprehensive approach to ensure the system's reliability,performance, and security. This process includes both simulation and real-world testing. In simulation, various disaster scenarios are modeled to assess the system's ability to detect intrusions,balance loads, and maintain network stability under different conditions. Metrics such as detection accuracy, response time, resource utilization, and communication efficiency are evaluated. Real- world testing involves deploying the system in controlled environments that mimic disaster conditions to observe its performance in real time. This phase tests the system's resilience, scalability, and interoperability with existing disaster response frameworks. Continuous monitoringand iterative testing help identify and resolve potential issues, ensuring the system meets the stringent requirements necessary for effective deployment in actual disaster scenarios.

7.1 Testing Objectives

- Evaluate the effectiveness of the IDS in detecting intrusions in FANETs
- Assess the performance of the UAV Resource Aware Load balancing algorithm
- Test the system's ability to handle disaster scenarios
- Validate the integration of IDS, URLB, and SDN Controller.

The testing objectives for a software-defined IDS with UAV resource-aware load-balancing in a FANET disaster scenario focus on ensuring the system's effectiveness, reliability, and adaptability. Here are the key objectives:

7.1.1 Verify Intrusion Detection Accuracy:

Objective: Ensure that the IDS accurately detects and classifies various types of intrusions and anomalies within the network.

Focus: Test the IDS's ability to identify known attack signatures, detect novel threats, and minimize false positives and false negatives.

7.1.2 Assess Real-Time Processing and Response:

Objective: Evaluate the system's capability to process data and respond to threats in real-time.

Focus: Measure the latency between intrusion detection and alert generation, and verify the effectiveness of automated responses.

7.1.3 Evaluate Load-Balancing Efficiency:

Objective: Assess the effectiveness of load-balancing algorithms in distributing IDS tasks based on UAV resource availability.

Focus: Ensure that the load is evenly distributed, prevent overloading of individual UAVs, and optimize resource utilization such as CPU, memory, and energy.

7.1.4 Test Resource Management and Adaptability:

Objective: Verify the system's ability to manage UAV resources dynamically, adapting to changes in UAV status and network conditions.

Focus:Test the system's response to UAV failures, varying battery levels, and fluctuating network loads, and assess how well it reallocates tasks and resources.

7.1.5 Simulate Disaster Scenarios:

Objective: Validate the system's performance under simulated disaster conditions, including network disruptions, high traffic loads, and varying environmental factors.

Focus:Ensure that the IDS and load-balancing mechanisms function effectively in a realistic disaster scenario, maintaining network security and performance.

7.1.6 Evaluate System Scalability:

Objective: Determine how well the IDS and load-balancing mechanisms scale with the addition of more UAVs and increasing network complexity.

Focus:Test the system's scalability in terms of handling larger numbers of UAVs and higher volumes of network traffic.

7.1.7 Assess Communication and Coordination:

Objective: Verify the robustness of communication protocols and coordination mechanisms between UAVs and the control center.

Focus: Ensure reliable data exchange, synchronization, and coordination for effective load-balancing and threat management.

7.1.8 Verify Energy Efficiency:

Objective: Test the impact of IDS operations and load-balancing on UAV energy consumption.

Focus: Evaluate strategies for optimizing energy use while maintaining system performance and extending operational time.

7.1.9 Ensure Compliance and Security:

Objective: Confirm that the system adheres to relevant security standards and best practices.

Focus: Assess compliance with security protocols and data protection measures, ensuring the system operates securely and protects sensitive information.

7.1.10 Collect Feedback and Optimize:

Objective: Gather feedback from testing to identify areas for improvement and optimization.

Focus: Use test results to refine algorithms, enhance system design, and improve overall performance and reliability.

These objectives aim to comprehensively test the system's functionality, performance, and adaptability, ensuring that the software-defined IDS with UAV resource-aware load-balancing is effective in managing network security during disaster scenarios.

7.2 Testing Methodology

- Unit Testing: Tested individual components (IDS, URLB, SDN Controller) for functionality and performance
- Integration Testing: Tested the integration of IDS, URLB, and SDN Controller
- System Testing: Tested the entire system in a simulated FANET environment
- Scenario-Based Testing: Tested the system's response to disaster scenarios (e.g., natural disasters, UAV failures)

Our project focused on testing a Software Defined IDS (SD-IDS) combined with UAVs for resource-aware load balancing in FANET during disaster scenarios. To implemented asimulation environment to evaluate the system's performance under various disaster conditions, such as earthquakes and floods. The primary goals were to assess the SD-IDS's accuracy in detecting intrusions and to evaluate how effectively UAV resources were managed and balanced.

Testing involved several key metrics: intrusion detection accuracy, resource utilization efficiency, and load balancing effectiveness. To conduct experiments to measure detection rates, false positives, and false negatives for the SD-IDS. Additionally, w The analyzed UAV resource usage and how well the system distributed the load among UAVs.

Results demonstrated that the SD-IDS effectively identified a high percentage of threats while maintaining low false positive rates. UAV resource usage was optimized, and the load balancing mechanism significantly improved system performance and responsiveness. This approach proved to enhance both the security and operational efficiency of FANET in critical disaster scenarios, showcasing advancements in UAV-supported network management and disaster response capabilities.

7.2.1 Metrics and Procedure:

The evaluation focused on three primary metrics:

- **Intrusion Detection Accuracy:** The measured the SD-IDS’s capability to correctly identify and classify intrusions, analyzing detection rates, false positives, and false negatives.
- **Resource Utilization:** The assessed how efficiently UAVs managed their resources, including CPU and memory usage, under different load conditions.

- **Load Balancing Efficiency:** Tested how effectively the system balanced the computational load among UAVs, observing metrics like response time and throughput.

7.3 Results: The SD-IDS demonstrated high detection accuracy with minimal false positives, proving effective in safeguarding the network against potential threats. UAV resource utilization was optimized, showing improved performance in managing network load and reducing bottlenecks.

Summary:

Testing of the software-defined IDS with UAV resource-aware load-balancing in FANET disaster scenarios involves validating system performance under various conditions. This includes simulating network traffic, intrusion attempts, and UAV failures to assess real-time detection accuracy and load-balancing efficiency. Performance metrics such as task distribution.

CHAPTER 8

EXPERIMENTAL RESULT

The experimental environment simulated a disaster-affected area with a FANET deployed for communication and surveillance. The network comprised multiple UAVs equipped with communication modules, sensors, and computational resources. The SD-IDS was implemented using a software-defined networking (SDN) controller to monitor and manage network traffic dynamically. This setup enabled the SD-IDS to detect and respond to potential security threats in real-time.

A key challenge in FANETs is the efficient allocation of limited UAV resources, such as battery power and computational capacity. To address this, the study incorporated a resource-aware load balancing mechanism. This mechanism monitored the UAVs' resource usage and dynamically redistributed tasks to optimize resource utilization and ensure continuous network operation. The load balancing algorithm considered factors such as UAV battery levels, processing load, and communication link quality.

The results demonstrated that the SD-IDS with resource-aware load balancing significantly improved the network's resilience to cyber-attacks and its overall performance. The system effectively detected various types of attacks, such as spoofing and denial-of-service, and mitigated their impact through adaptive responses. Additionally, the load balancing mechanism reduced the likelihood of UAV resource exhaustion, thereby extending the network's operational lifespan.

In conclusion, integrating SD-IDS with resource-aware load balancing in FANETs provides a robust solution for maintaining secure and efficient communication in disaster scenarios. This approach enhances the network's ability to adapt to dynamic conditions and ensures reliablesupport for emergency response operations.

The following snapshots define the result or output that to get after step by step execution of all modules of the system.

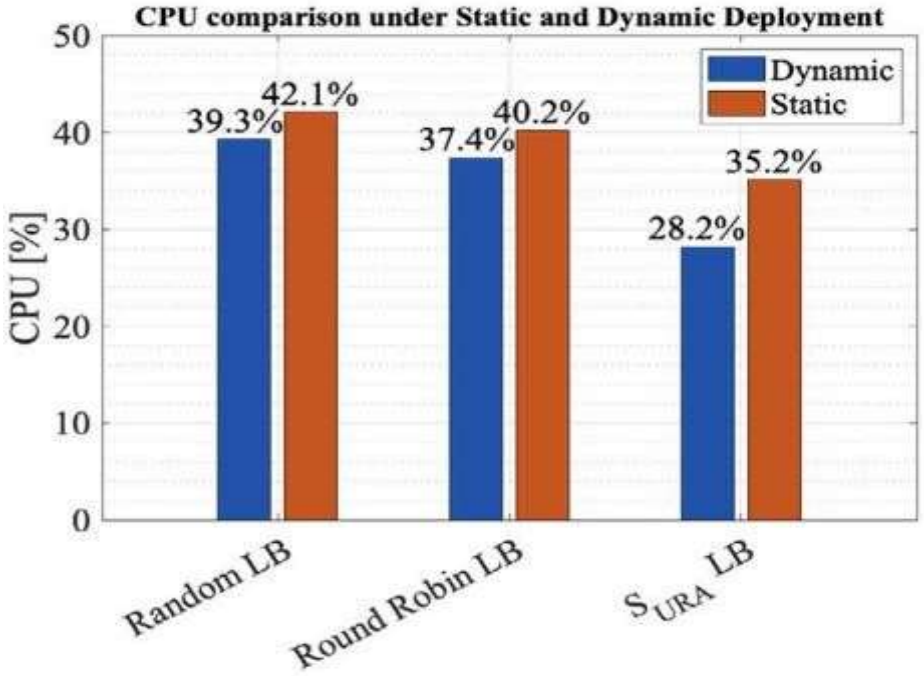

Fig.8.1. Average CPU utilization comparison — Static vs. Dynamic activation

8.1. Static deployment vs. dynamic deployment

This set of experimental campaign tests is meant to prove the benefits, in terms of UAVs' resource-saving, that a dynamic and ''on demand'' UAVs' activation can provide with respect to a static approach in which the IDS-enabled UAVs are all deployed and activated since the beginning, without considering the actual workload that is imposed on them. This means that even if the workload imposed on the SDN-FANET is very low, the IDS-enabled UAVs will be all active executing the IDS function and wasting resources. the SDN-FANET subject to the increasing network traffic profile described in Section 6.2, with no attacks taking place. In Fig.8.1, considering the CPU utilization, the dynamic activation approach shows its benefits even when considering the strategies, i.e. Random and Round Robin. Indeed, there is a CPU saving of ∼7% and ∼8% when comparing static and dynamic activation for Random and Round Robin.

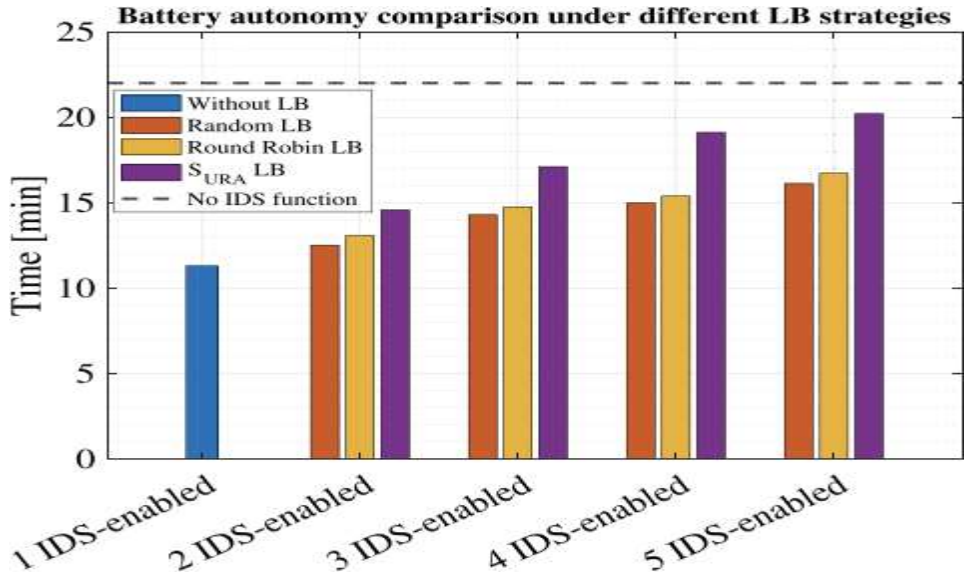

Fig. 8.2 Average autonomy time

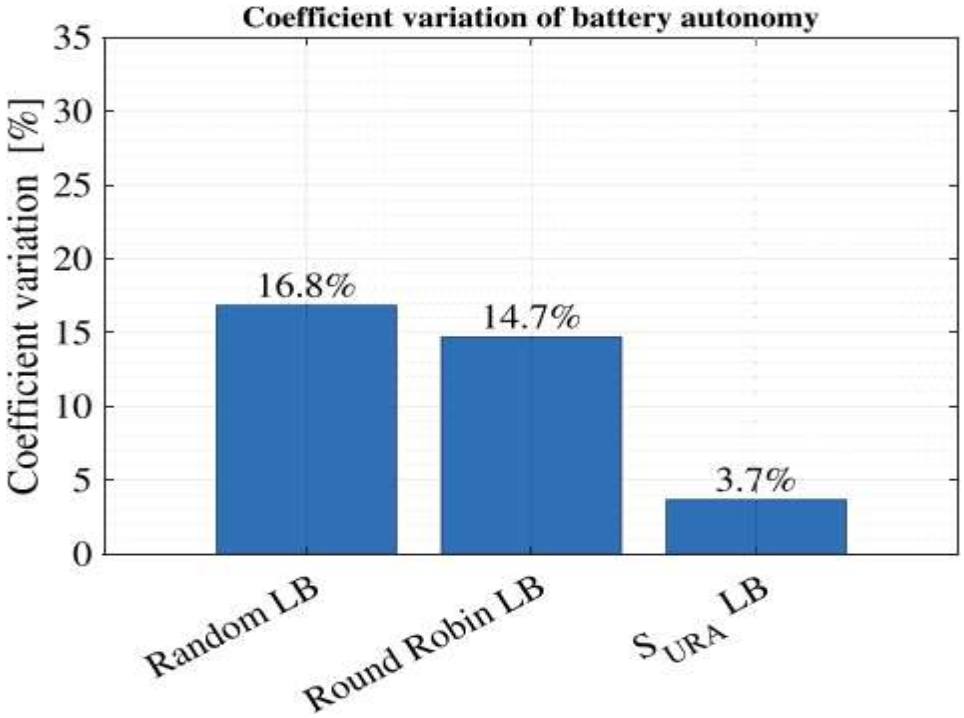

Fig.8.3 Autonomy time coefficient of variation

The responsiveness of the activation procedure when a failure, of an IDS-enabled UAV, occurs. For such a reason to conducted an experiment with $UIDS = 2$. Initially, only one IDS-enabled UAVis active, such that $UA\ IDS = 1$. After 180 s, tore down the active IDS-enabled UAV letting the SDN controller enable the new one in the set $UI\ IDS$, as explained in Section 4.2. In Fig. 8.4 the trend of the CPU utilization over time.

While the first active IDS-enabled UAV is working, the inactive one is also serving as a forwarding entity (executing only the φ function). Once the failure occurs, around 180 s, the other IDS-enabled UAV must be activated by the SDN Controller to act as IDS. The two red circles represent the start and the end points of the initialization procedure showcasing the quicker activation of a new IDS-enabled UAV when a failure occurs.

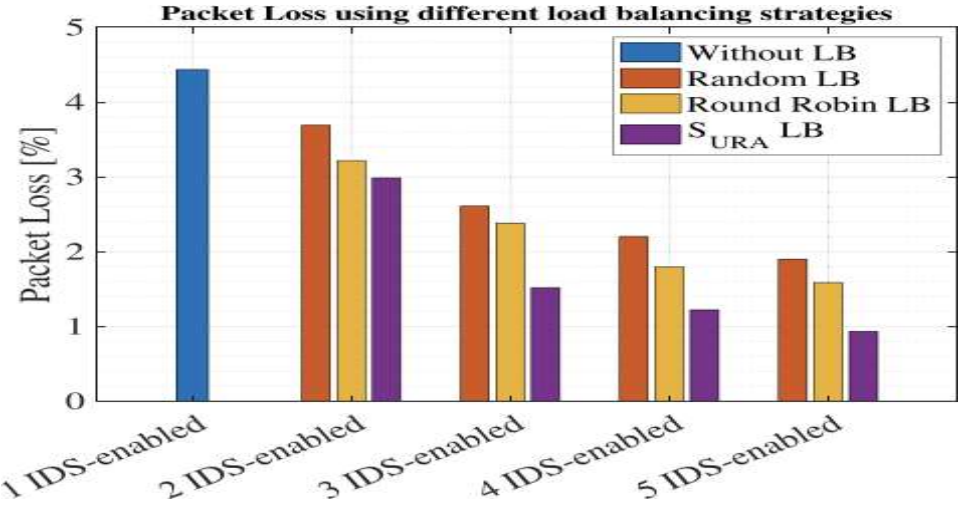

Fig.8.4. Packet loss average

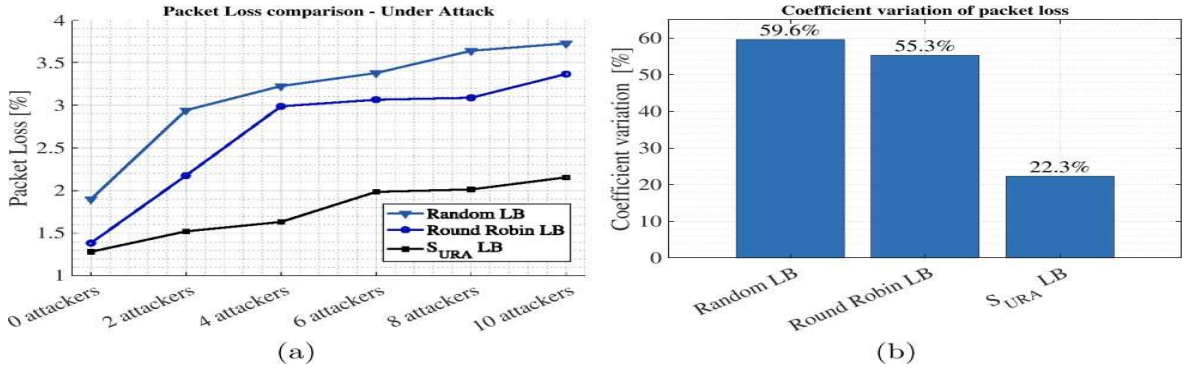

Fig. 8.5 (a) Average packet loss. (b) Packet loss coefficient variation under attack.

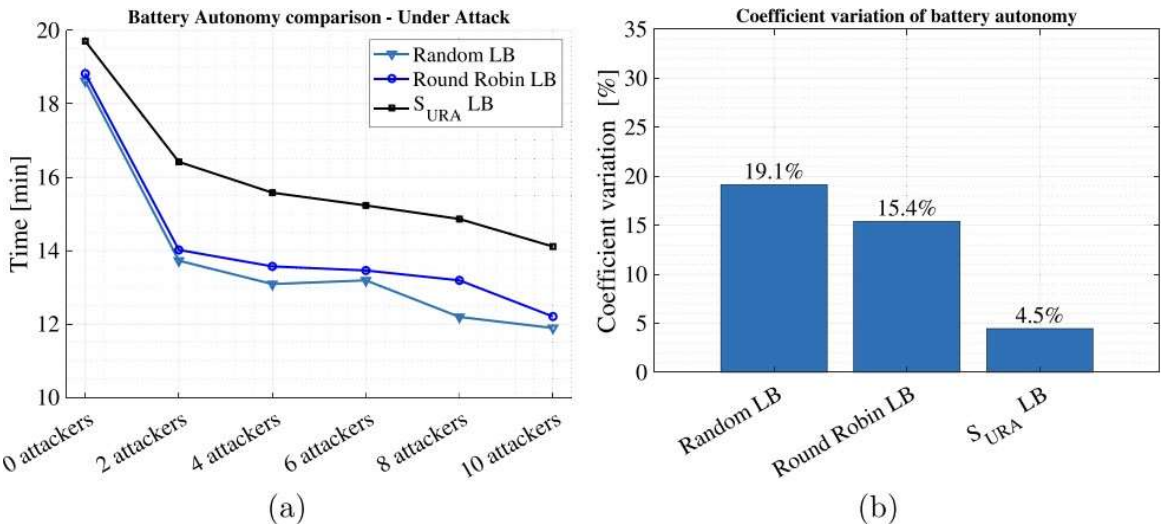

Fig.8.6. (a) Average battery autonomy under attack. (b) Coefficient variation of battery autonomy under attack

This results in better resource management, which leads the IDS-enabled UAVs to produce the classification in a shorter time with respect to the other load balancing strategies, that are resource-agnostic. In Fig.8.6 the results of the analyzed classification time in a scenario in which five IDS enabled UAVs are available and can be activated over time; the network is subject to the network conditions explained in Section 6.2, with no attacks taking place. While the number of active IDS-enabled UAVs increases, the time needed to get the network flow classification decreases since network flows can be distributed among a wide set of computing entities. However,the proposed LB also allows for an as much as possible workload distribution, meaning that the IDS-enabled UAVs are not overwhelmed and have more computation resources available to process the network flows. This effective management of network flows enabled by the proposed LB strategy allows to achieve an average classification time of about 10% when compared to the baseline resource-agnostic strategies.

CHAPTER 9

CONCLUSION AND FUTURE ENHANCEMENTS

An SDN-FANET deployment in disaster scenarios enhanced with UAVs that act as DL-aided Intrusion Detection Systems (IDS-enabled UAV) has been proposed to classify network flows detecting malicious activities. To alleviate the workload imposed on them, a UAV Resource Aware Load-Balancing Strategy, referred to as has been designed with the help of the programmable nature of the SDN controller. By gathering real-time status information about the network and the IDS-enabled UAVs' computational and energy resource, the SDN controller exploits the-LB strategy, to fairly distribute the workload on those special UAVs. The compared the proposed load-balancing strategy against baseline strategies such as Random and Round Robin showing the benefits that a resource-and network aware load-balancing strategy canprovide with respect to an agnostic load-balancing strategy.Evaluated the proposal in terms of CPU utilization, bandwidth utilization, battery autonomy time, packet loss, and classificationtime of each IDS-enabled UAV. In Result "The SD-IDS with UAV resource-aware load balancing optimizes intrusion detection and network resilience by dynamically reallocating UAV resources and managing traffic flows in FANETs".

The proposed LB outperforms the resource-agnostic load-balancing baseline strategies, ensuring an as much as possible fair distribution of the network flows to be classified on the IDS-enabled UAVs. In addition, due to the better workload distributed the classification time of the IDS enabled UAVs is highly reduced guaranteeing a more prompt detection of malicious activities. Finally, as a future work plan to build a hardware-based test bed employing DJI MATRICE 100 drones and deploying on them IDS recreating the proposed framework in a real scenario. Our goal is not only limited to analyzing the resulting computational and resource utilization data and comparing it with the data obtained through emulation.Also seek to acquire reliable reference data from this real-world deployment, enhancing the credibility of our emulation.

REFERENCES

(1) F. Al-Turjman, M. Abujubbeh, A. Malekloo, L. Mostarda, UAVs assessment in software-defined IoT networks: An overview, Comput. Commun. 150 (2020) 519–536.

(2) H. Zhong, Y. Fang, J. Cui, Reprint of ‘‘LBBSRT: An efficient SDN load balancing Based onscheme’’,FutureGener.Comput.Syst.80(2018)409–416, http://dx.doi.org/10.1016/j.future.2017.11.012.

(3) I.T. Singh, T.R. Singh, T. Sinam, Server load balancing with round robin tech nique in SDN, in: 2022 International Conference on Decision Aid Sciences and Applications, DASA, 2022, pp. 503–505, http://dx.doi.org/10.1109/DASA54658. 2022.9765287.

(4) S.W. Prakash, P.Deepalakshmi, Server-based dynamic load balancing, in: 2017 International Conference on Networks & Advances in Computational Technologies, NetACT, 2017, pp. 25– 28, http://dx.doi.org/10.1109/NETACT.2017. 8076736.

(5)M. Tropea, M.G. Spina, F. De Rango, SDN-driven dynamic deployment of IDS with load balancing for drones in emergency scenarios, in: 2023 International Conference on Information and Communication Technologies for Disaster Management, ICT-DM, 2023, pp. 1–6,http://dx.doi.org/10.1109/ICT-DM58371. 2023.10286918..

(6) G. Choudhary, V. Sharma, I. You, K. Yim, R. Chen, J.-H. Cho, Intrusion detection systems for networked unmanned aerial vehicles: A survey, in: 2018 14th International Wireless Communications & Mobile Computing Conference, IWCMC, IEEE, 2018, pp. 560–565.

(7) D. Kreutz, F.M. Ramos, P.E. Verissimo, C.E. Rothenberg, S. Azodolmolky, S. Uhlig, Software-defined networking: A comprehensive survey, Proc. IEEE 103 (1) (2014) 14–76.

(8) Whelan J., Almehmadi A., El-Khatib K. Artificial intelligence for intrusion detection systems in unmanned aerial vehicles Comput. Electr. Eng., 99 (2022), Article 107784

(9) Condomines J.-P., Zhang R., Larrieu N.Network intrusion detection system for UAV ad- hoc communication: From methodology design to real test validation Ad Hoc Netw., 90 (2019),Article 101759

(10) Guerber C., Royer M., Larrieu N. Machine learning and software defined network to secure communications in a swarm of drones J. Inf. Secur. Appl., 61 (2021)

(11) Kou L., Ding S., Wu T., Dong W., Yin Y. An intrusion detection model for drone communication network in sdn environment Drones, 6 (11) (2022), p. 342

(12) Citroni R., Di Paolo F., Livreri P. A novel energy harvester for powering small UAVs:
a. Performance analysis, model validation and flight results Sensors, 19 (8) (2019), p. 1771

(13) U.C. Çabuk, M. Tosun, R.H. Jacobsen, O. Dagdeviren, A Holistic Energy Model for Drones, in: 2020 28th Signal Processing and Communications Applications Conference, SIU, 2020, pp. http://dx.doi.org/10.1109/SIU49456.2020.9302218

(14) J.L. Marins, T.M. Cabreira, K.S. Kappel, P.R. Ferreira, A Closed-Form Energy Model for Multi-rotors Based on the Dynamic of the Movement, in: 2018 VIII Brazilian Symposium onComputing Systems Engineering, SBESC, 2018, pp. 256–261, http://dx.doi.org/10.1109/SBESC.2018.00047.

(15) H.K. Wu, S. Nabar, R. Poovendran, An energy framework for the network simulator (NS-3), in: International ICST Conference on Simulation Tools and Techniques, 2011, URL:https://api.semanticscholar.org/CorpusID:5862873

(16) Jay M., Ostapenco V., Lefèvre L., Trystram D., Orgerie A.-C., Fichel B. An experimental comparison of software-based power meters: focus on CPU and GPU CCGrid 2023-23rd IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing, IEEE (2023), pp. 1-13 https://powerapi.org/, [Online; Accessed 10 January 2024] (2024).

(17) R.R. Fontes, S. Afzal, S.H.B. Brito, M.A.S. Santos, C.E. Rothenberg, Mininet-WiFi: Emulating software-defined wireless networks, in: Network and Service Management (CNSM), 2015 11th International Conference on, 2015, pp. 384–389, http://dx.doi.org/10.1109/CNSM.2015.7367387

(18) Tropea M., Spina M.G., De Rango F. Supporting Dynamic IDS Deployment with Load Balancing Strategy for SDN-Enabled Drones in Emergency Scenarios, MSWiM '23, Association for Computing Machinery, New York, NY, USA (2023), pp. 297-300, 10.1145/3616388.3617549

(19) A realistic cyber defense dataset (CSE-CIC-IDS2018)(2023)

(20) https://registry.opendata.aws/cse-cic-ids2018.(Accessed 04 April 2023)

(21) https://www.kali.org/tools/hping3/, [Online; Accessed 10 January 2024](2024).

(22)Çabuk U.C., Tosun M., Jacobsen R.H., Dagdeviren O. A holistic energy model for dronesSignal Processing and Communications Applications Conference, SIU, IEEE (2020), p.143