UNIVERSITÀ
DEGLI STUDI
DELL'AQUILA

Software Engineering for Autonomous Systems

Project Documentation: Self-Adaptive Network Traffic Management and Intrusion Mitigation System Using SDN

Submitted to:
Professor Davide Di Ruscio

Submitted by:
Minase Mengistu (297864)
Vinay Sanga (297853)
Thinakone Louangdy (297862)

February, 2025

Contents

1 Project Description **2**

2 Goals of the System **2**

3 Functional Requirements **2**

4 Non-Functional Requirements **3**

5 System Components **3**
5.1 Managed Resources . . . 3
5.2 Sensors and Effectors . . . 3
5.2.1 Sensors . . . 4
5.2.2 Effectors . . . 4

6 Autonomic Manager **4**
6.1 Architectural Pattern . . . 4
6.1.1 Self-Awareness and Context-Awareness . . . 5
6.2 Key Aspects of Self-Adaptation . . . 5
6.3 Adaptation Timing . . . 6
6.4 Adaptation Techniques . . . 6
6.5 Adaptation Control . . . 6
6.6 Decision Function . . . 6
6.6.1 Machine Learning-Based Classification . . . 6
6.6.2 Rule-Based Traffic Control . . . 8
6.6.3 Real-Time Network Adaptation . . . 8
6.6.4 Network Statistics Logging . . . 8
6.6.5 Adaptive Thresholding (Temporary Fix for Demo) . . . 9
6.7 MAPE-K Implementation . . . 9

7 System Architecture and Sequence Diagram **9**
7.1 System Architecture . . . 9
7.2 Sequence Diagram . . . 10

8 Technologies Used **11**

9 Conclusion **12**

1 Project Description

The DDoS Detection and Mitigation System is an autonomous software solution designed to safeguard SDN-based networks against Distributed Denial-of-Service (DDoS) attacks. Leveraging the MAPE-K (Monitor, Analyze, Plan, Execute, and Knowledge) feedback loop, the system continuously monitors network traffic, analyzes flow patterns for anomalies, formulates mitigation strategies, and enforces adaptive flow rules to minimize the impact of DDoS attacks.

The project is built on a modular architecture comprising microservices for monitoring, analysis, planning, and execution. The core functionalities are containerized and managed via Docker Compose for scalability and ease of deployment. A virtualized network environment is created using Mininet, where synthetic DDoS traffic is generated to test the system.

2 Goals of the System

The primary goal of the system is to ensure the availability and performance of the network under potential DDoS attacks. Specific objectives include:

1. Detect Anomalous Traffic: Identify DDoS traffic using real-time monitoring and machine learning classification.
2. Mitigate Threats Dynamically: Apply flow rules dynamically to limit or eliminate malicious traffic while prioritizing legitimate traffic.
3. Adapt to Changing Network Conditions: React to network anomalies in real time to maintain stability and efficiency.
4. Log Network Performance: Maintain historical logs of traffic patterns for analysis and optimization.

3 Functional Requirements

Table 1: Functional Requirements of the DDoS Detection System

Identifier	**Name**	**Description**
DDOS-FR001	Traffic Monitoring	Continuously monitor traffic using OpenFlow switches and the Ryu SDN controller.
DDOS-FR002	Traffic Classification	Use an ML-based model to classify traffic as benign, suspicious, or DDoS.
DDOS-FR003	Dynamic Mitigation	Formulate and enforce flow rules to handle suspicious and malicious traffic.
DDOS-FR004	Logging and Reporting	Log traffic patterns and classification results to InfluxDB for analysis.

4 Non-Functional Requirements

Table 2: Non-Functional Requirements of the DDoS Detection System

Identifier	Name	Description
DDOS-NFR001	Scalability	The system is scalable due to the capabilities of the Ryu SDN Controller, which efficiently handles high flow volumes and multiple switches through its optimized OpenFlow implementation and asynchronous processing.
DDOS-NFR002	Modularity	Ensure independent operation of the Monitor, Analyzer, Planner, and Executor.
DDOS-NFR003	Reliability	Maintain consistent performance under heavy DDoS traffic.

5 System Components

5.1 Managed Resources

The managed resources of the system include:

- **Virtualized Network:** The network is created using **Mininet**, providing a flexible and customizable environment for testing network behaviors. While the default topology follows a **tree structure** (depth of 2, fanout of 2), the system is **not limited to this configuration**. Users can modify the topology based on their needs, supporting various network structures such as:
  - **Tree topology** (default)
  - **Linear topology** (simple chain of switches and hosts)
  - **Mesh topology** (interconnected nodes for redundancy)
  - **Custom-defined topologies** (fully programmable using Mininet's Python API)

  This flexibility allows testing in different network environments, from **small-scale simulations** to **large-scale SDN deployments**.
- **Synthetic Traffic:** The system generates both normal and malicious network traffic. DDoS attacks, such as **SYN flooding**, **ICMP flooding (ping flood)**, and **HTTP request flooding**, are simulated using tools like `hping3` and `ping -f`. Additionally, normal traffic (e.g., HTTP requests) can be introduced to measure the impact of an attack on legitimate traffic.

5.2 Sensors and Effectors

The system relies on a combination of sensors and effectors to monitor network activity and enforce traffic control policies.

5.2.1 Sensors

The system uses the Ryu SDN controller to monitor network flow statistics in real time. The key sensor components include:

- **Flow Statistics Collection:** OpenFlow-enabled switches collect and report flow statistics to the controller. These include:
  - Packet count
  - Byte count
  - Flow duration
  - Source and destination MAC/IP addresses
- **DDoS Traffic Detection:** The Analyzer module processes network flow data and classifies traffic patterns into benign, suspicious, or DDoS categories using a machine learning model.

5.2.2 Effectors

The system uses effectors to apply mitigation strategies based on the Analyzer's classification results. The key effectors include:

- **Dynamic Flow Rule Enforcement:** The Executor module interacts with the Ryu SDN controller to modify flow rules dynamically. These include:
  - **Traffic Prioritization:** Assigns higher priority to benign traffic.
  - **Rate Limiting:** Reduces the bandwidth allocated to suspicious traffic to minimize potential harm.
  - **Packet Dropping:** Completely blocks identified DDoS traffic to protect the network.
- **Automated Adaptation:** The system continuously updates and refines its flow rules based on new observations and classification results, ensuring that mitigation strategies remain effective against evolving threats.

6 Autonomic Manager

6.1 Architectural Pattern

The autonomic manager in the **Self-Adaptive Network Traffic Management System** is responsible for dynamically monitoring, analyzing, planning, and executing mitigation strategies to protect the SDN environment from DDoS attacks. It follows a **self-adaptive system** approach, integrating **machine learning-based classification**, **rule-based decision making**, and **real-time network adaptation** to ensure optimal performance.

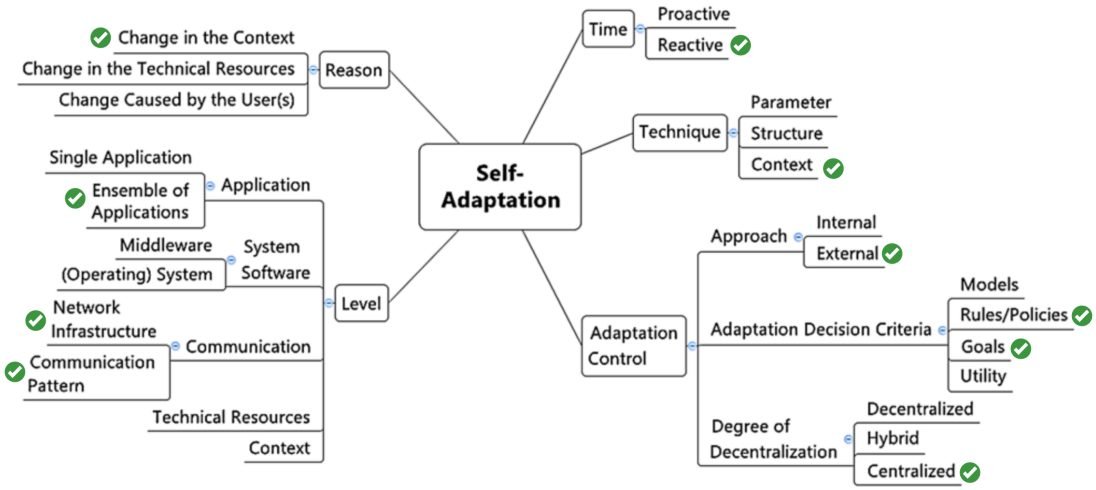

Figure 1: Architectural Pattern.

6.1.1 Self-Awareness and Context-Awareness

- **Self-awareness:** The system continuously monitors network flow statistics, detects abnormal patterns, and applies mitigation measures automatically.
- **Context-awareness:** The system considers environmental factors such as total network traffic, active switches, and byte/packet flow rates before deciding on adaptation strategies.

6.2 Key Aspects of Self-Adaptation

The autonomic manager operates according to the following adaptation principles:

- **When to adapt?** The system adapts when a DDoS attack or anomalous network activity is detected.
- **Why adapt?** To ensure network availability and protect legitimate traffic from congestion.
- **Where to adapt?** Adaptation occurs in the SDN controller, where OpenFlow rules are dynamically modified.
- **What kind of adaptation?** The system employs **parameter adaptation** (adjusting QoS levels, rate limits) and **structural adaptation** (modifying flow rules).
- **Who performs the adaptation?** The **Planner** determines adaptation strategies, and the **Executor** enforces them via the SDN controller.
- **How is adaptation performed?** Using a **hybrid decision-making approach**, combining machine learning, predefined policies, and real-time network monitoring.

6.3 Adaptation Timing

- **Reactive adaptation:** The system responds immediately to detected anomalies by updating SDN flow rules.

6.4 Adaptation Techniques

The autonomic manager uses a combination of adaptation techniques:

- **Parameter adaptation:** Dynamically adjusts bandwidth allocation, flow priorities, and rate limits.
- **Structural adaptation:** Modifies OpenFlow rules to prioritize, rate-limit, or block traffic.
- **Context adaptation:** Takes real-time network conditions into account when determining mitigation actions.

6.5 Adaptation Control

- **External adaptation approach:** The adaptation logic is implemented independently from the underlying network infrastructure.
- **Centralized adaptation logic:** The **Planner** acts as a global decision-maker for network-wide policy enforcement.
- **Decision-making criteria:** The system uses **machine learning-based classification** combined with predefined **rule-based policies** to determine traffic handling strategies.

6.6 Decision Function

The autonomic manager employs a **hybrid decision function** integrating **machine learning classification**, **rule-based traffic control**, and **real-time monitoring**.

6.6.1 Machine Learning-Based Classification

- A **Random Forest classifier** trained on the **CIC-DDoS2019 dataset** is used to classify network flows.
- The model analyzes key flow features, including:
  - Flow Duration
  - Packet Rate
  - Byte Rate
  - Packet Count
  - Byte Count

- The trained model is deployed in the **Analyzer** to classify incoming traffic as **Benign, Suspicious, or DDoS**.

The model's performance was evaluated using the CIC-DDoS2019 dataset, with a focus on key metrics such as accuracy, precision, recall, and F1-score. The following plots highlight the results:

Confusion Matrix The confusion matrix in Figure 2 illustrates the model's classification performance. It shows the number of true positives, true negatives, false positives, and false negatives:

- **True Positives (TP):** Correctly classified DDoS attack traffic.
- **True Negatives (TN):** Correctly classified benign traffic.
- **False Positives (FP):** Benign traffic misclassified as DDoS.
- **False Negatives (FN):** DDoS traffic misclassified as benign.

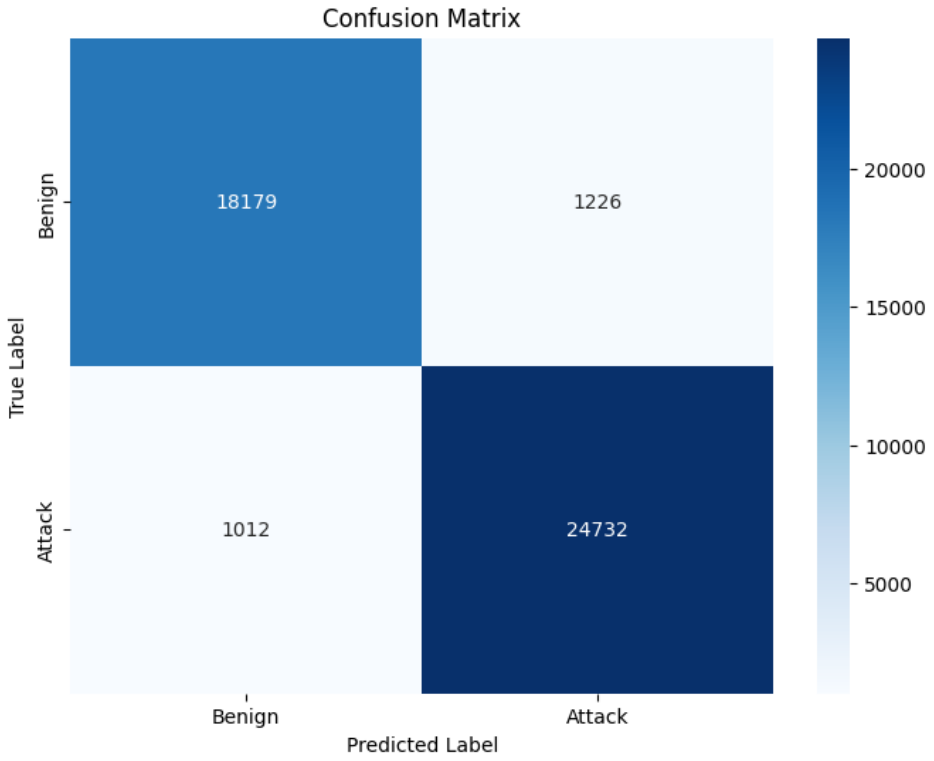

Figure 2: Confusion Matrix of the Random Forest Model

Precision-Recall Curve The precision-recall curve in Figure 3 evaluates the trade-off between precision and recall across different thresholds. The model achieves an **average precision (AP) score of 0.94**, demonstrating its ability to distinguish between benign and attack traffic effectively.
These results highlight the effectiveness of the Random Forest classifier in providing accurate and reliable traffic classification, which is critical for supporting the Planner's decision-making process.

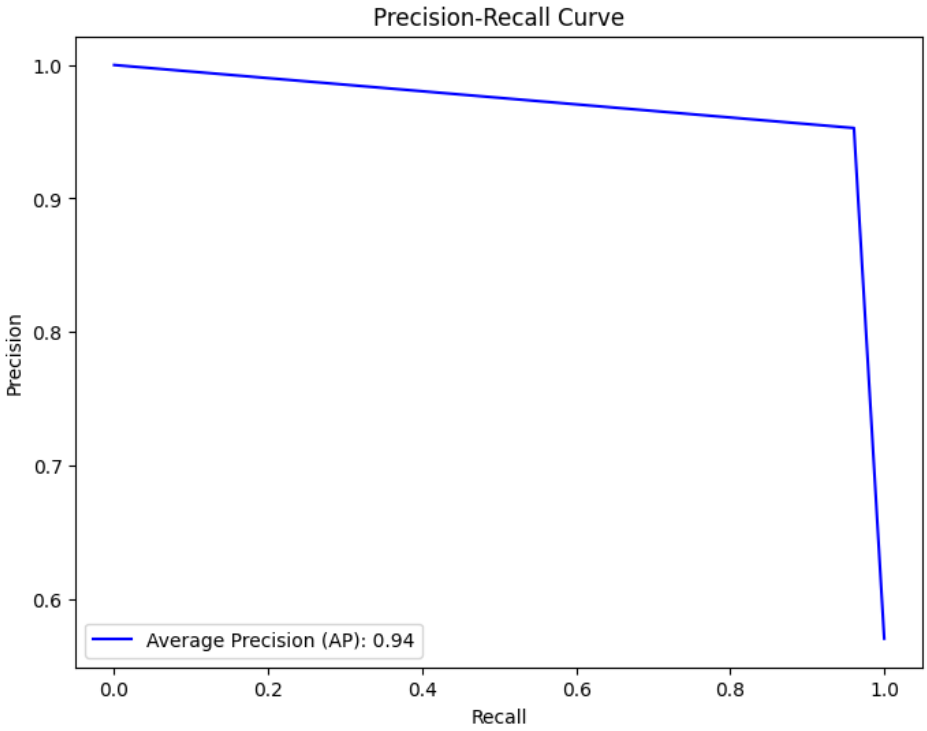

Figure 3: Precision-Recall Curve for the Random Forest Model

6.6.2 Rule-Based Traffic Control

To complement machine learning classification, the system applies predefined traffic handling policies:

- **Benign traffic**: Forwarded normally with high-priority QoS.
- **Suspicious traffic**: Rate-limited to prevent possible congestion.
- **DDoS traffic**: Completely dropped to mitigate attack impact.

6.6.3 Real-Time Network Adaptation

- The **Planner** decides on the appropriate mitigation strategy based on classification results.
- The **Executor** applies OpenFlow rule modifications in the Ryu SDN Controller.

6.6.4 Network Statistics Logging

To assess mitigation effectiveness, the system retrieves and logs **real-time network statistics** from the Ryu SDN Controller before and after mitigation:

- **Planner (Pre-Mitigation)**: Requests total packet count, total byte count, and the number of active switches before applying mitigation.
- **Executor (Post-Mitigation)**: Retrieves updated network statistics after mitigation to assess impact.

- Both components store these statistics in **InfluxDB** for further analysis.

6.6.5 Adaptive Thresholding (Temporary Fix for Demo)

During testing, the initial model classified nearly all flows as **Benign** due to dataset differences. To temporarily address this issue for demonstration purposes:

- An **adaptive thresholding mechanism** was introduced to dynamically adjust classification boundaries.
- Instead of using fixed thresholds, the system calculated **T1 and T2** based on real-time observations.

This mechanism was only a temporary fix and is not intended as a long-term adaptation strategy.

6.7 MAPE-K Implementation

The autonomic manager follows the **MAPE-K loop**, ensuring self-adaptation:

- **Monitor (Ryu Controller):** Continuously collects network flow statistics.
- **Analyze (Machine Learning Model):** Classifies network traffic using a trained **Random Forest classifier**.
- **Plan (Decision Engine):** Determines the optimal mitigation strategy based on classification results.
- **Execute (Flow Rule Enforcer):** Enforces mitigation actions by modifying SDN flow rules.
- **Knowledge (Historical Data Storage):** Stores network statistics in **InfluxDB** for future analysis.

7 System Architecture and Sequence Diagram

7.1 System Architecture

The architecture of the Self-Adaptive Network Traffic Management System is designed to follow the principles of the MAPE-K loop, ensuring dynamic and efficient DDoS mitigation in an SDN environment. The system consists of the following components:

- **Mininet (Managed Resource):** Simulates the SDN network with hosts and switches to generate and test network traffic.
- **Ryu SDN Controller (Monitor):** Collects flow statistics from switches and forwards them to the Analyzer for classification.
- **Analyzer:** Implements a trained Random Forest model for traffic classification (Benign, Suspicious, or DDoS).

- **Planner:** Determines the appropriate mitigation strategy based on classification results and retrieves network statistics from the Ryu controller.
- **Executor:** Enforces the mitigation strategy by modifying OpenFlow rules in the SDN controller.
- **InfluxDB (Knowledge Base):** Stores general network statistics (e.g., total packets, bytes, active switches) before and after mitigation.
- **Grafana:** Provides a visualization interface for real-time and historical network data.

Figure 4 illustrates the system architecture and the flow of information between components:

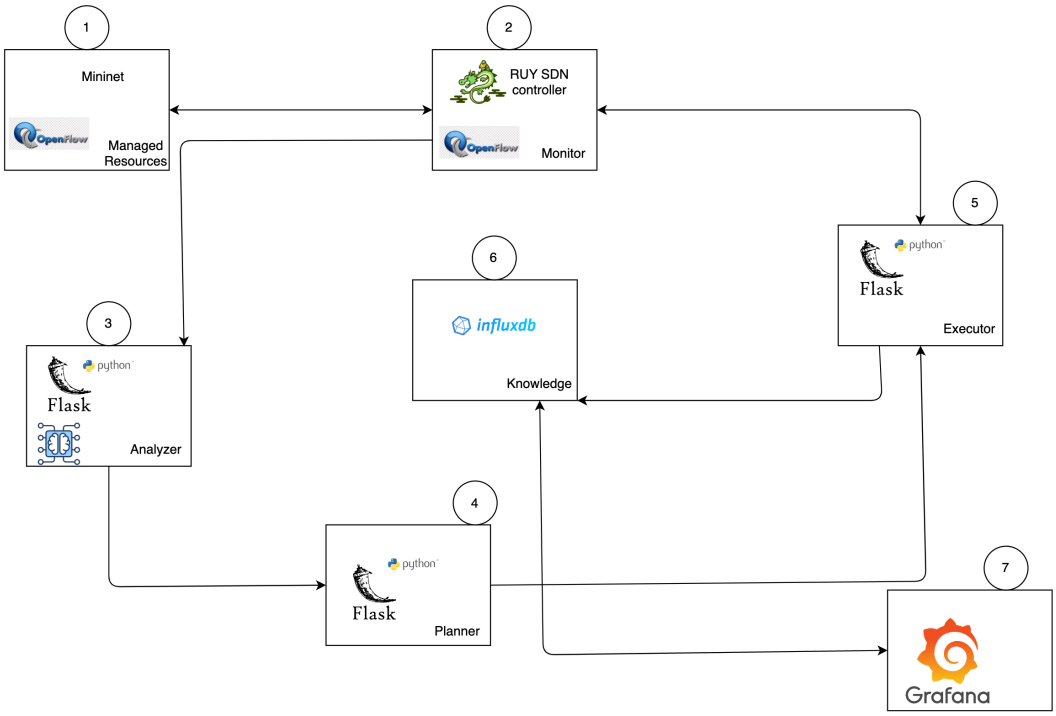

Figure 4: System Architecture Diagram

7.2 Sequence Diagram

The sequence diagram shown Figure 5 demonstrates the interactions between components in the system, following the MAPE-K loop:

- **Monitoring:** The Ryu SDN controller collects real-time flow statistics.
- **Analysis:** The Analyzer classifies traffic using the trained Random Forest model.
- **Planning:** The Planner retrieves pre-mitigation statistics, determines mitigation strategies, and stores them in the Knowledge Base.

- **Execution:** The Executor enforces the planned mitigation by modifying OpenFlow rules and logs post-mitigation statistics.

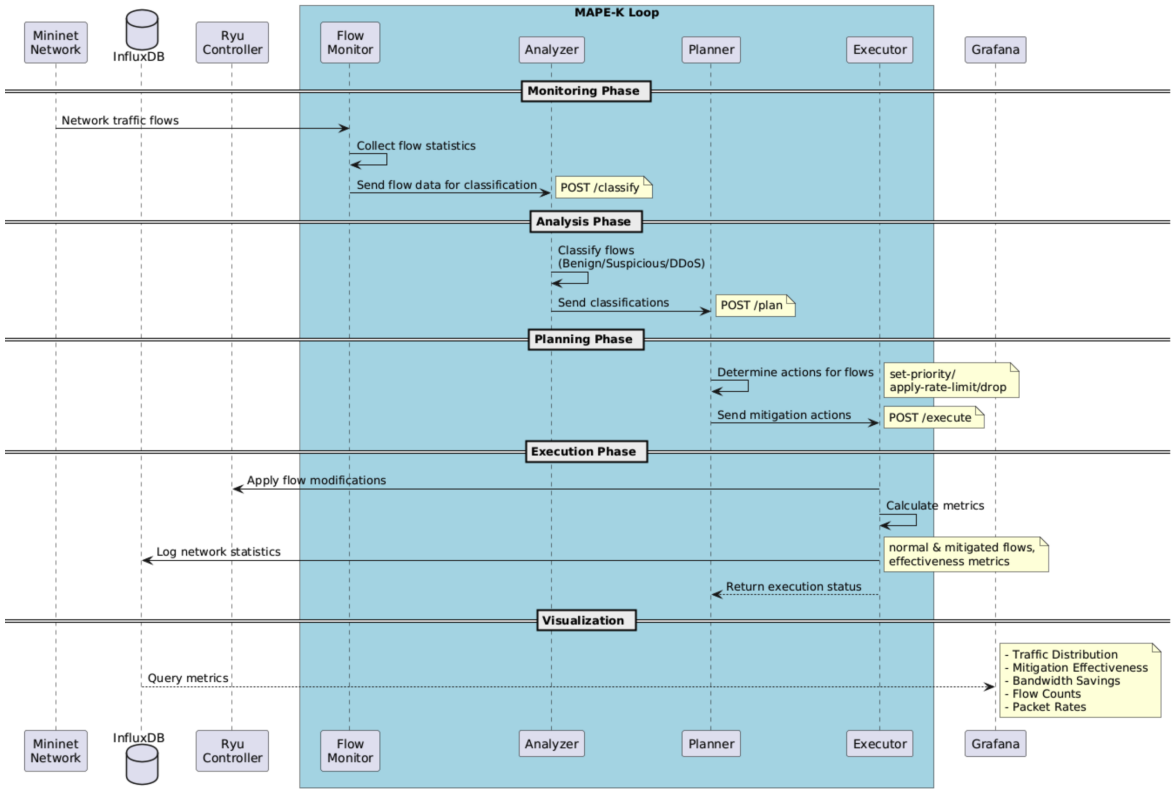

Figure 5: Sequence Diagram

8 Technologies Used

Mininet

Ryu SDN Controller

Flask

Docker

InfluxDB

Grafana

Python

OpenFlow

9 Conclusion

In this report, we presented a comprehensive overview of the Self-Adaptive Network Traffic Management System designed to mitigate DDoS attacks in an SDN environment. The system leverages a hybrid approach that combines machine learning-based classification, rule-based decision making, and real-time adaptation to ensure network resilience and availability. By implementing the MAPE-K loop, the autonomic manager dynamically monitors, analyzes, plans, and executes mitigation strategies at the network infrastructure level.

Key contributions of the system include:

- Real-time traffic classification using a Random Forest model trained on the CIC-DDoS2019 dataset.
- Centralized decision-making via the Planner, which enforces parameter, structural, and context-based adaptations.
- Integration of real-time monitoring and post-mitigation logging through the Ryu SDN controller's REST API.
- Deployment of a scalable and modular architecture utilizing Mininet, Docker, and Flask-based microservices.

In general, this project highlights the potential of self-adaptive systems in modern SDN environments to address security challenges dynamically and effectively.