RASA-Driven Assistant for Real-Time Monitoring and Management of Distributed SDN Controllers for Campus Networks

Gonella Siva Sai Surya Prakash, Chandan Chatragadda, Vinodha K, and Animesh Giri

PES University, Bengaluru 560100, India

gonellasurya2005@gmail.com, chandan365c@gmail.com, vinodhak@pes.edu
animeshgiri@pes.edu

Abstract. Managing large-scale networks, such as campus networks in the schools or universities requires continuous monitoring, rapid fault detection and real-time visibility. Traditional Software Defined Networking (SDN) tools rely on manual configurations and static dashboards and thus, lack the flexibility and the usability needed for easy management and quick decision making. This paper presents a smart RASA-based conversational AI framework integrated with a distributed ONOS SDN controller cluster to improve its observability, usability, and proactive network management. The system enables operators to interact with the network using natural language, allowing them to query the network state, manage flows, and execute actions like host blocking. A real-time monitoring of controller health, link and switches ensures high availability, while a Flask-based alert server pushes alerts to the user interface proactively. Integration with InfluxDB and Grafana provides dynamic visualization of traffic patterns and network performance. The system also includes a real-time anomaly detection, classification mechanisms and rapid mitigation actions. The experimental evaluation on Mininet-based campus network topologies demonstrates the improved fault response time, better observability and scalability for real-world deployments

Keywords: Software-Defined Networking (SDN), ONOS (Open Network Operating System), RASA, Conversational AI, Distributed Controllers, Anomaly Detection, Real-Time Monitoring, Campus Networks.

1 Introduction

Modern educational institutions and corporate campuses are becoming increasingly dependent on large-scale, high performance networks to support research, communication, and daily operations. These networks must handle a growing number of devices, applications, and users, all while ensuring seamless connectivity and security. Managing these networks requires not only technical resilience but also real-time insights, fault tolerance, and natural interaction mechanisms. Traditional Software-Defined Network (SDN) tools, while powerful, often do not provide the flexibility or user-friendly design needed to manage complex network environments effectively, especially when dealing with distributed controller setups like ONOS clusters.

Campus networks typically demand high availability, rapid fault recovery, and proactive anomaly detection to prevent disruptions that could affect critical activities such as online learning, research collaborations, or business operations. However, most existing SDN platforms rely on command-line interfaces or static dashboards, which limit the operator's ability to react swiftly during critical events such as link failures, traffic spikes, or security breaches. These limitations not only increase the operational burden on network administrators but also heighten the risk of prolonged downtimes or undetected threats. These issues highlight the urgent need for a smarter, more innovative, and human-friendly network management solution that can adapt to the dynamic nature of campus environments.

Our project addresses the limitations of traditional SDN management by introducing a RASA-powered conversational AI assistant that allows users to control and monitor a distributed ONOS controller cluster through natural language. Unlike conventional tools that rely heavily on scripts and command-line interaction, our assistant simplifies the operational workflow by accepting user queries and commands in everyday language, making it accessible even to those without deep technical expertise.

This conversational interface helps solve the problem of limited accessibility and slow fault response in large-scale campus networks. With support for querying network status, identifying controller or link failures, and inspecting real-time topology data such as hosts, devices, paths, flows, etc., the system allows network administrators to act faster and with greater clarity. By offering a user-friendly interaction model, our project bridges the gap between technical complexity and operational usability in SDN environments, ultimately reducing the time and effort required for network management.

To ensure timely communication and fault transparency, our system integrates a lightweight Flask backend that enables the assistant to push real-time alerts to a dedicated Graphical User Interface (GUI). This mechanism allows the system to inform users about network issues such as unreachable devices, traffic surges, or suspicious host behavior without requiring manual refresh or polling.

One of the major issues in distributed controller setups is ensuring high availability. Our solution tackles this challenge through continuous controller health monitoring and automated failover handling. When a controller node becomes inactive, the system dynamically reroutes management queries to other available ONOS instances in the cluster, ensuring uninterrupted operation and improving the fault tolerance of the network as a whole. This approach minimizes service disruptions and maintains network performance even under failure conditions.

This project, though developed as a research-oriented prototype, is designed with scalability and real-world applicability in mind. By integrating multiple technologies—including SDN via ONOS, RASA-based conversational AI, Grafana for data visualization, and Flask for real-time communication—the system creates a cohesive platform for intelligent and efficient SDN management. Furthermore, the modular design allows for easy expansion, such as incorporating additional AI features or supporting larger network topologies.

Our approach demonstrates how automation and natural interfaces can significantly enhance the operational agility of campus-scale networks. Through features like anomaly detection from port statistics, real-time host blocking, and traffic visualization

dashboards, the system presents a forward-thinking model that can be adapted for deployment in universities, such as proactive security suggestions, voice integration, and support for additional SDN controllers. This not only improves efficiency but also paves the way for more intuitive network management in the future.

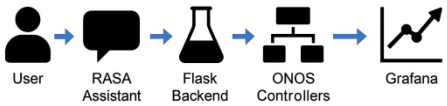

Fig. 1. Conversational SDN Monitoring Architecture Flow

A. Problem Statement

Campus networks in places like universities or large companies are becoming more complex and spread out. Managing these modern Software-Defined Networks (SDNs) often requires real-time monitoring, quick fault handling, and automation. But most traditional SDN tools need manual setup, don't provide a user-friendly experience, and can slow down decision-making, especially when there are multiple controllers involved. There is a need for a smarter and more efficient SDN management system that:

- Lets network operators interact with the network and manage it using simple natural language input.
- Detects controller or link failures in real-time and pushes alerts to the user interface.
- Identifies unusual traffic patterns and classifies them as possible anomalies and logs them.
- Helps take quick action through commands like blocking suspicious hosts.
- Shows a visual dashboard with port stats and live network status for better understanding.

This project tries to solve these problems by combining a RASA-based chatbot with a distributed ONOS controller setup. It also includes real-time alerting, anomaly detection, and visual monitoring using a Flask backend and Grafana dashboard. The goal is to create an easy-to-use, scalable, and smart SDN assistant that's a good fit for managing campuswide networks

2 Background

A. Software-Defined Networking (SDN)

Software-Defined Networking (SDN) introduces a new paradigm by separating the control plane from the data plane, enabling centralized management of network devices through programmable controllers. This separation is designed to improve scalability and flexibility, while simplifying network administration, which is particularly useful in large scale environments like university campuses and enterprise networks. Despite these advantages, managing such networks requires real-time monitoring, quick fault recovery, and proactive anomaly detection to ensure uninterrupted services such as communication, research, and daily operations. Traditional SDN management tools often rely on command-line interfaces or static dashboards. These methods can be

slow, require deep technical expertise, and are less responsive to dynamic events such as link failures, sudden traffic spikes, or potential security breaches.

B. ONOS

The Open Network Operating System (ONOS) is a widely adopted SDN controller known for its scalability and support for distributed, multi-controller environments. ONOS clusters coordinate across multiple controller instances to provide fault tolerance, high availability, and improved network performance.

Key mechanisms in ONOS include state synchronization and mastership election, which ensure that network devices are always managed by an active controller. The coordination among distributed controllers is supported by Atomix, a framework that handles synchronization and consistency across the cluster.

While ONOS provides powerful capabilities, managing distributed controllers often becomes complex, particularly without intuitive interfaces or automation. This can overwhelm administrators and slow down response times in fast-changing network conditions.

C. RASA Framework

The RASA framework is an open-source platform for building conversational AI chatbots, utilizing AI and ML for self-training with minimal data, as noted in [6]. Its modular architecture, comprising RASA Core (dialogue management) and RASA NLU (natural language understanding), supports integration with various systems, including SDN controllers like ONOS [5]. RASA Core manages conversation flow using a tracker object to store dialogue states, while RASA NLU classifies intents and extracts entities from user inputs, converting them into numerical representations for processing [8].

RASA's custom actions, defined in the actions.py file, allow tailored responses to user intents, such as triggering ONOS API calls for network management tasks [6]. The framework's HTTP APIs enable compatibility with diverse programming environments, making it suitable for our project's needs. We chose RASA for its open-source nature, self-hosted deployment, and flexibility in handling custom actions, which facilitate seamless interaction between the chatbot and the SDN infrastructure [8].

3 Related Works

A. Intent-based Networking (IBN)

Intent-based Networking (IBN) is a transformative approach to network management that automates configuration by allowing users to define high-level objectives, which the system translates into actionable network policies. As described in [5], IBN enables network operators to specify desired outcomes (e.g., ensuring connectivity or blocking specific traffic) without detailing the exact sequence of commands, hence simplifying management and enhancing scalability. This paradigm focuses on the intended network state rather than the procedural steps to achieve it, making it robust for large-scale networks where low-level states change rapidly.

IBN leverages artificial intelligence (AI) and machine learning (ML) to enhance decision-making and automation. According to [3], AI/ML techniques are crucial in

evolving SDN frameworks by assisting human operators or automating control processes based on defined intents. These advancements enable proactive monitoring, anomaly detection, and adaptive responses, aligning network behavior with user expectations.

B. Distributed Controllers and Campus-Wide Networks

Distributed controller architectures have become essential for managing large-scale and mission-critical networks such as those found in universities, research campuses, and corporate environments. Unlike single-controller setups, distributed controllers provide scalability and high availability by replicating state across multiple nodes. This ensures that failures in one controller do not disrupt overall operations, allowing seamless failover and reduced downtime. Previous studies have shown that campus networks benefit from distributed Software-Defined Networking (SDN) architectures by achieving improved fault tolerance, faster decision making, and greater coverage of heterogeneous devices spread across multiple locations.

The Open Network Operating System (ONOS) has emerged as a leading distributed controller platform in this space. While ONOS is well-suited for campus-scale deployments, the complexity of monitoring and controlling multiple controllers often requires deep technical expertise. Traditional tools still rely on manual configurations or static dashboards, which can be slow to react to events such as link failures or traffic anomalies. These limitations have led to the exploration of more intuitive management approaches. One promising direction is the use of conversational AI assistants, which can reduce complexity and support proactive campus-wide network management.

C. Related Studies

Several studies provide insights into SDN management and conversational AI integration, aligning with our project's objectives. In [4], the authors compare single and multicontroller SDN architectures using ONOS and Mininet, focusing on performance metrics like latency and throughput. Their findings highlight the benefits of multi-controller setups in fault tolerance and scalability. This supports our choice of a distributed ONOS cluster. However, the study lacks exploration of higher-level automation or user driven interfaces, a gap that our RASA-based assistant addresses.

The work in [2] introduces the Intent Monitor and Reroute (IMR) service for ONOS, enabling dynamic routing based on real-time flow statistics. This service supports external control, aligning with our goal of integrating intelligent modules like RASA for intent-driven management. However, like much of the related work, this system has mainly been validated in emulated environments. Our prototype similarly operates in an emulated setup but extends the approach by integrating conversational AI for more intuitive, real-time management.

In [3], a framework combining cognitive machine reasoning and RASA-based NLP with ONOS controllers is proposed for intent-based orchestration. This system, tested in a simulated environment, demonstrates the feasibility of conversational AI in SDN but lacks distributed controller integration, a key feature of our work. Similarly, [5] presents a RASA-based chatbot for packet-optical networks, simplifying operations through natural language but focusing on specific use cases without scalability validation.

Unlike these works, our prototype integrates RASA with a distributed ONOS cluster, demonstrating the feasibility of a user-friendly approach for campus networks. By combining real time monitoring, anomaly detection, and custom actions, our system improves accessibility and operational efficiency, addressing limitations in previous studies.

4 Proposed Methodology

The methodology proposed aims to build a scalable and intelligent network management framework for campus-wide SDN deployments. The system combines the power of conversational AI with a distributed SDN controller cluster, enabling real-time visibility, anomaly detection, and network control using natural language. The methodology is organized into several modules, each contributing to a more intuitive, fault tolerant, and responsive management system.

A. Conversational AI Layer and Intent Recognition

To simplify the way users interact with the SDN network, we have implemented a RASA-based conversational AI. This conversational chatbot allows users to perform actions such as viewing devices, adding flows, or blocking hosts using simple voice or text commands.

- Intents like get devices, add flow, and block host are extracted from the user's message using RASA's NLU engine.
- Slot values are retrieved and passed to custom action functions that format and trigger REST API calls to ONOS controllers. The JSON payloads are created as per the requirement.

This layer serves as a user-friendly front of the entire architecture and significantly reduces the complexity of interacting with the network by enabling natural language inputs.

B. Distributed Controller Communication

The system is designed to work with a cluster of ONOS controllers, ensuring high availability and fault tolerance. Atomix handles distributed coordination and state synchronization among controller instances in this setup.

- A controller selection logic checks which ONOS node has mastership over the target device.
- If a controller becomes unreachable, requests are automatically redirected to a healthy node with the required mastership.

This ensures smooth operation and reduces the risk of single-point failures.

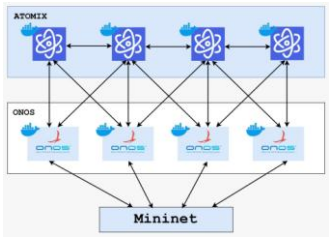

Fig. 2. ONOS Multi-Controller Architecture [4]

C. Real-Time Monitoring and Alerting

To maintain network health, the system continuously monitors both controller and link statuses and alerts users in real-time.

- A controller health monitor checks the availability of each ONOS node and notifies if any controller becomes unreachable. The specific group of switches which were connected to the controller which failed are automatically reassigned to an active controller in the cluster.
- In the same way, a function tracks if any switch or link in the network goes down unexpectedly.
- Alerts are pushed to a lightweight Flask-based server and are periodically fetched by the frontend, allowing timely visibility for the user.

This monitoring adds a layer of resilience and proactive fault handling and alerting.

D. Anomaly Detection using Port Statistics

The system also includes a rule-based anomaly detection engine that classifies suspicious traffic behaviors using ONOS port statistics.

- It polls port-level statistics like packet counts at regular intervals.
- Any sudden spike or irregular traffic pattern is flagged and classified using heuristics.
- Detected anomalies are logged to a file and are pushed to the GUI as alerts.

This module adds intelligence to the assistant, helping it go beyond passive control and into active detection.

E. Mitigation with Host Blocking

When anomalies or threats are detected, the system can respond quickly with mitigation actions.

- Using natural language commands, the user can trigger a block on a suspicious host.
- The system, on confirmation from the user, pushes a high priority flow rule to drop all packets from that host using ONOS's flow API.

This provides a fast and controlled action to limit potential damage from compromised devices.

F. Visual Dashboard and Statistics Visualization

To complement the conversational interface, Grafana is used to visualize network metrics in a dynamic dashboard.

- The assistant periodically exports port statistics and traffic data from ONOS into InfluxDB.
- Grafana connects to InfluxDB and pulls this data and displays real-time graphs of traffic trends, providing insights into traffic patterns and anomalies.

This module enhances the observability of the network and supports better decision making.

G. Flask Alert Server

A minimal Flask backend is implemented to store and serve alerts generated by the assistant's monitoring functions.

- Alerts are stored temporarily and served through a /get alerts endpoint.
- A background cleaner periodically clears the memory to avoid overloading the server.

This helps the frontend receive alerts in near real-time without the need for complex backend infrastructure.

5 System Design

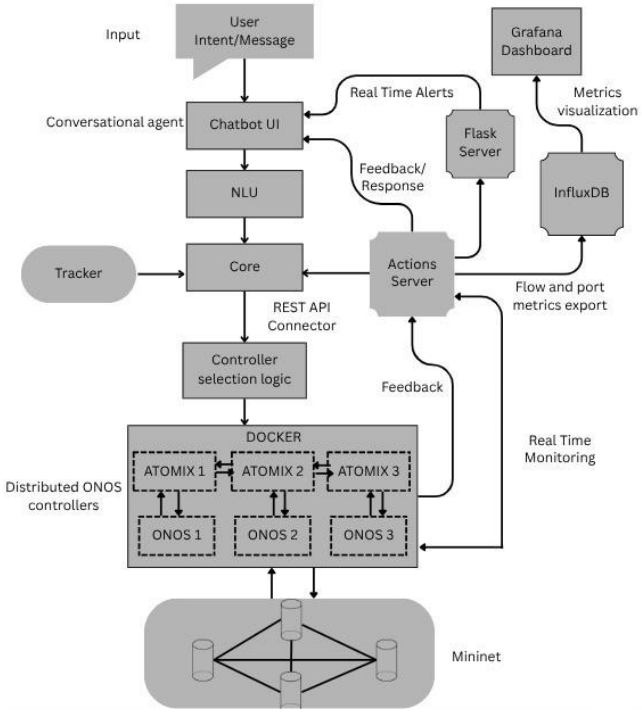

Fig. 3. System Architecture

6 Implementation Details

This section presents the implementation details of our project that focuses on building an intelligent and conversational SDN management system for campus-wide deployments. The system integrates a RASA-based assistant with a distributed ONOS SDN controller setup, enabling users to monitor and control the network through natural language inputs. The following are the key technologies used in our implementation:

A. Frontend

- RASA Chatbot: Interfaces which processes the natural language queries from

the user and sends them to the actions server.
• HTML, CSS, JavaScript: To design the web-based GUI for the RASA chatbot.
• Grafana: For visual dashboards to monitor real-time port statistics and anomalies.

B. Backend
• RASA Actions Server: Executes backend logic in response to user intents, including communication with ONOS via APIs.
• Flask: Lightweight server to handle alerts storage for the frontend to access.

C. Network Layer / Core System
• ONOS (SDN Controller): Manages the network by installing flow rules and maintaining the network state.
• Atomix: Enables coordination and synchronization among distributed ONOS controller instances.
• Mininet: Simulates network topologies and generates controlled traffic/anomalies for testing.
• Operating System: Ubuntu 24.04 LTS D. Data Storage & Visualization
• InfluxDB: Stores network performance metrics (e.g., port statistics) in a time-series format.
• Grafana: Takes the InfluxDB data to generate dynamic dashboards for visual monitoring.

E. Deployment Environment

ONOS controllers and Atomix were deployed using Docker, allowing a five node distributed cluster to be set up with minimal configuration. RASA, Flask, InfluxDB and Grafana were deployed directly on Ubuntu 24.04. Docker simplified the cluster management during testing.

F. Testing Setup

Mininet was used to emulate distributed campus-like network topologies. Controlled traffic patterns were generated to simulate anomalies such as traffic spikes, link failures and host misbehavior. These scenarios were used to test and validate the functioning of the system.

G. Code Snippets
1. send to healthy controller(): Finds the master controller (if required) and sends request to healthy controller.
2. Monitor network functions:
    monitor controllers(): Periodically monitors the controller health and alerts the user if any controller goes down.
    monitor topology health(): Periodically monitors the switches and links and alerts the user if any of them go down.
3. monitor anomalies(): Continuously monitors the network for any unusual traffic and alerts the user if any suspicious traffic spikes are detected.

4. block host(): Issues commands to block or unblock a suspicious host, which installs a flow rule to drop traffic from that host.
5. ui alert server.py: A mini flask server which temporarily stores real-time alerts generated by monitoring scripts and are pulled by the frontend GUI.

7 Results and Declaration

The proposed system was implemented to enable intelligent, real time, and fault-tolerant management of campus-wide SDN environments. Our architecture combines a RASA-based assistant, ONOS controller cluster, and real-time monitoring modules and daemons to address the needs of scalable SDN operations. Below are the key results and observations:

1. Intent-based network management through Conversational AI: The RASA assistant successfully handled natural language commands like add flow, block host, and get devices, allowing seamless interaction with the SDN network without requiring low-level technical input from the user, reducing the need for in-depth syntax expertise. This enables faster decision making and easier control over the network.
2. Controller health monitoring and Failover handling: The controller monitoring function was able to detect controller failures in real time. If the master controller was unreachable, the system automatically checked other available nodes to find a suitable backup controller, ensuring high availability. The state synchronization, new mastership election, and cluster membership is handled by Atomix, a distributed controller coordination framework, which ensures a smooth operation, synchronization, and coordination of the distributed controllers.
3. Real-time Anomaly Detection and Classification: The anomaly monitoring module accurately flagged and classified unusual traffic patterns based on port statistics. Alerts were logged into a file and were also pushed to the UI, providing timely feedback to the network administrator.
4. Quick Mitigation through Host Blocking: The assistant could block traffic from suspicious hosts using a simple command. This action was implemented by injecting drop flow rules through the ONOS REST API into the appropriate switches in the network topology, showing the system's ability to quickly respond to network threats.
5. Flask-based Alert Server and Frontend Integration: The lightweight Flask server reliably stored and served alert messages generated by the monitoring modules. The RASA frontend pulled these alerts in real-time, offering users visual feedback for better decision-making. A background cleaner periodically cleared the memory to avoid overloading the server and ensured a smooth operation.
6. Visualization with Grafana Dashboard: Port statistics collected from ONOS controllers were organized and successfully visualized using Grafana, offering graphical insights into live traffic data. This dashboard supported proactive monitoring and simplified network planning and scaling.

8 Conclusion and Future work

In this project, we designed and implemented a smart distributed SDN management framework tailored for campus networks. Our system integrates a conversational AI assistant built using RASA with a multi-controller ONOS SDN controller setup, enabling real-time network monitoring, fault detection, anomaly classification, and intent-based flow control through natural language interaction.

This system was introduced to enhance user awareness and reduce response time. To achieve this goal, we implemented continuous health monitoring for controllers, switches, and links, real-time alerting through a lightweight Flask backend, and anomaly detection based on port-level statistics. A simple host blocking mechanism was also included to allow immediate mitigation of suspicious traffic. The integration of Grafana for visual monitoring also improves the observability of the overall network.

This solution addresses some of the key challenges in traditional distributed SDN management, such as lack of interactivity, slow and manual troubleshooting, and poor detection of faults by offering a more intelligent, intuitive, and modular alternative. This design lays a strong foundation for future production-grade SDN automation tools.

A. Future Work

We have identified several future directions to further improve and enhance this project:

- Adding ML-based Intrusion Detection and prediction models for advanced anomaly classification and intrusion detection, with the potential to predict faults before they impact the network allows for a better security and management of the campus networks.
- Deploying the chatbot using distributed RASA instances behind a load balancer (e.g., NGINX or HAProxy) can prevent it from becoming a single point of failure, increasing the availability and reducing down times.
- Making the Alert Granularity finer, such as providing location-specific alerts, like building-level or switch-level diagnostics, can help the network administrator make a quicker and more detailed and targeted response.
- Adding Role-Based Access in the assistant (e.g., admin, auditor, etc) can help with better security and network segmentation.
- Adding Intent Chaining (multiple actions in one command) can increase the ease of performing complex operations and improve usability.

References

1. Chaudhary, R., Aujla, G.S., Kumar, N., Chouhan, P.K.: A comprehensive survey on software-defined networking for smart communities. Int. J. Commun. Syst. 37(1), (2024). https://doi.org/10.1002/dac.5296
2. Ali, U.: Performance Comparison Of SDN Controllers In Different Network Topologies. Research Square (2024). https://doi.org/10.21203/rs.3.rs-4276322/v1
3. Asif, M., Khan, T.A., Song, W.-C.: Leveraging Cognitive Machine Reasoning and NLP for Automated Intent-Based Networking and e2e Service Orchestration.

IEEE Access 13, 65684–65698. https://doi.org/10.1109/ACCESS.2025.3298751

4. Abraham, A.L., Hrishikesh, B.H., Atehar, M.T., Hegde, B.N., Giri, A.: ONOS SDN Framework: Assessing the Impact of Single and Multi-Controller Architectures on Network Efficiency. In: Advances in Intelligent Computing and Communication. Springer (2024). https://doi.org/10.1007/978-981-97-1323-3_32
5. Cesila, C.H., Pinto, R.P., Mayer, K.S., Escallon-Portilla, A.F., Mello, D.A.A., Arantes, D.S., Rothenberg, C.E.: Chat-IBN-RASA: Building an Intent Translator for Packet-Optical Networks based on RASA. In: IEEE Conf. Netw. Softwarization, pp. 1–5 . https://doi.org/10.1109/NetSoft57336.2023.10167245
6. Srivastava, P., Agrawal, A., Tiwari, V., Singh, A.K.: Building a Chatbot on a Closed Domain using RASA. In: Proc. 3rd Int. Conf. Adv. Comput., Commun. Control Netw. (ICACCCN), pp. 667–671. ACM, Greater Noida, India (2021). https://doi.org/10.1145/3443279.3443308
7. Ghobadi, A., Moosavi, S.R., Shojafar, M., Buyya, R.: RL Employing a Multi Objective Reward Function for SDN Routing Optimization. IEEE Trans. Netw. Service Manag. 18(4), 4093–4104. https://doi.org/10.1109/TNSM.2023.10392368
8. Setiawan, E.I., Santoso, J., Langgeng, Y.S.H.: Chatbot for University Students Major Determination with Rasa Framework. In: Proc. 27th Int. Comput. Sci. Eng. Conf. (ICSEC) (2023). https://doi.org/10.1109/ICSEC59635.2023.10329749