Mininet Remastered:Developing a New User-Friendly GUI for CLI-Based Mininet Emulator

I. Abstract

Large companies and organizations should look for more affordable and efficient ways to rapidly develop their networks, as well as better and simpler ways to administer their networks. Software-Defined Networking (SDN) is a new method of managing networks that allows for automated, programmatically efficient network setup to enhance monitoring and performance. It is used with virtualization to allow networking devices to be reconfigured into manageable software. This paper explores SDN topology created using Mininet emulator . Mininet is a popular network emulator offers a command-line interface (CLI) for building network topologies, but what we are facing is the widespread adoption of the CLI-based solution is impeded by the fact that users who lack extensive networking experience may find it intimidating. In order to overcome the usability issues with Mininet's CLI, we developed a new and unique Graphical User Interface (GUI) specifically designed for SDN topology development, which we called it "Mininet Remastered". Mininet Remastered is offering a clear, visual depiction of network components and their connections, seeks to expedite the process of emulation and creation of SDN topologies.

II. Introduction

An innovative approach to network management, software-defined networking (SDN) allows for more flexible and effective network configuration and operation. Conventional networks use hardware, such as switches and routers, to manage traffic, which can be rigid and difficult to maintain. By separating the control plane from the data plane, SDN enables programmable network configurations and centralized network intelligence. This separation simplifies the management of network resources, improves scalability, and accelerates the deployment of new applications and services. Central to the SDN paradigm is the use of controllers, and one of the most prominent examples is OpenDaylight. OpenDaylight, an open-source project under the Linux Foundation, provides a modular platform that supports various networking protocols and technologies, making it a versatile tool for network automation and orchestration. By enabling the dynamic adjustment of network policies and the efficient allocation of network resources, OpenDaylight significantly enhances overall network performance.

To effectively develop and test SDN solutions, network professionals often turn to Mininet, a network emulator that plays a crucial role in this ecosystem. Mininet creates a realistic virtual network environment, complete with hosts, switches, and links, all running real kernel and user code. This capability allows developers and researchers to experiment with network designs and configurations in a controlled, cost-effective manner before deploying them in a live environment. Together, SDN, OpenDaylight, and Mininet form a powerful combination, advancing the research, development, and deployment of next-generation networking solutions.

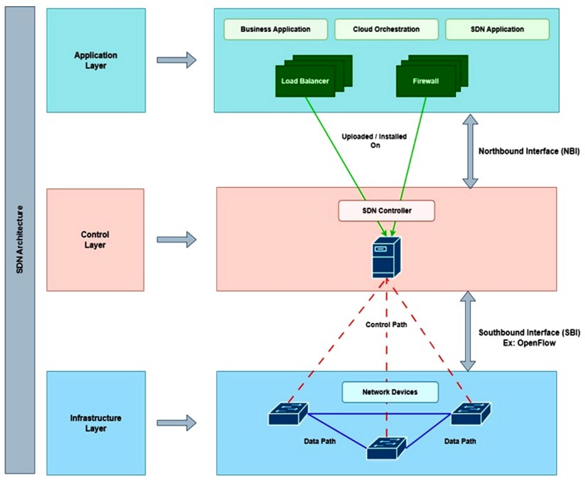

Fig. 1: SDN Architecture

In recent history, Software-Defined Networking (SDN) research has emerged as a significant field, offering transformative approaches to centralized and programmable network management. However, conveying SDN topologies effectively in a pictorial format within research papers brings unique obstacles, affecting clarity, comprehension, and overall impact.[4]

- **Diversity of Structural Elements:** The architecture of SDN topology comprises several components, including controllers, switches, routers, and end-hosts, each serving distinct functions inside the network. As SDN networks become increasingly intricate, visual representation poses significant challenges, particularly when topologies feature both horizontal and vertical layers. For instance, SDN controllers function at a control layer that is distinct from data plane devices, resulting in a layered topology that can be difficult to represent in a single diagram. Additionally, it is crucial to convey the unique attributes and interactions of each device clearly to prevent misinterpretation, a task that becomes more complex when limited to a constrained visual format. Illustrating elements such as packet flow, control commands, and device interactions often requires a level of complexity that may obscure the fundamental aspects of the architecture, potentially leaving readers confused or misinformed. Striking the right balance between abstraction and the retention of essential details in one or two comprehensive diagrams is particularly challenging.

- **Difficulties with Tools:** While tools such as Mininet have become increasingly popular in the realm of Software-Defined Networking (SDN) research for the emulation and testing of network topologies, they exhibit certain limitations regarding the generation of publishable visual representations. Mininet is proficient in creating emulated environments; however, it lacks the capability to export diagrams in formats that are appropriate for academic dissemination. Consequently, researchers frequently depend on supplementary software tools, including Graphviz or Microsoft Visio, to produce diagrams that strike a balance between simplicity and the necessary detail to convey essential components and their interrelations. Nevertheless, these tools often do not provide built-in support for SDN-specific elements, compelling researchers to manually design or modify symbols to accurately depict controllers, switches, and other devices. Furthermore, conventional diagramming software may impose restrictions on customization options tailored to the unique geometric requirements of SDN, complicating the representation of distinctive configurations, particularly in highly adaptive network models. Such limitations can lead to visual inaccuracies, thereby undermining the overall effectiveness of the research paper.
- **Dynamic Flow Representation:** The defining characteristic of Software-Defined Networking (SDN) is its ability to modify data flows dynamically in response to varying network conditions and established policies. This flexibility presents a considerable challenge for static visual representations. In contrast to conventional networks, where routing tends to be more stable, the programmability of SDN facilitates real-time, adaptive modifications that are challenging to encapsulate in a singular, static diagram. Researchers often depend on flow diagrams, which are inherently inadequate for illustrating the dynamic nature of SDN. For instance, showcasing adaptive flows may necessitate multiple diagrams or sequential images, but incorporating such elements can add complexity and disrupt the overall coherence of the paper. The inability to visually represent real-time modifications risks misrepresenting the flexibility of SDN, potentially leading to a reduced appreciation of its fundamental benefits among readers. These dynamic features are crucial for grasping the full potential of SDN, and effectively capturing them continues to be a significant challenge in the realm of SDN visualization.
- **Lack of Standardized Symbols:** Traditional networking diagrams have historically benefited from a set of standardized symbols for components such as routers, switches, and servers, which fosters consistency and ease of recognition. However, SDN currently lacks a universally accepted set of symbols, particularly for newer or distinctive elements like the SDN controller, which plays a critical role in the architecture. This lack of standardization compels researchers to create their own representations, often leading to visual inconsistencies that may confuse or mislead the audience. For example, some researchers might depict an SDN controller using a generic rectangle or circle, while others may opt for a more abstract shape, resulting in variability that complicates comparisons across studies. Furthermore, these custom or modified symbols may not clearly convey their intended function or role within the SDN framework, thereby reducing the interpretability of the diagrams. The establishment of a universally recognized set of symbols for SDN would significantly improve the clarity and comprehensibility of research diagrams.

III. Mininet Emulator

Mininet is a versatile and widely-used network emulator that provides an invaluable tool for network professionals, researchers, and developers. Its primary strength lies in its ability to create a realistic virtual network environment, enabling the emulation of large-scale networks on a single machine. This capability is particularly beneficial for testing and developing Software-Defined Networking (SDN) applications, where real-world testing can be prohibitively expensive and logistically complex. [2]

Mininet achieves this by using lightweight virtualization to run many instances of virtual hosts, switches, and links, all using real kernel and user space code. This setup allows users to create a complex network topology with minimal resource overhead, facilitating rapid prototyping and testing. The emulated network behaves similarly to a physical network, making it an ideal platform for experimenting with new network protocols, configurations, and SDN controllers.

For SDN research, Mininet is often paired with SDN controllers like OpenDaylight. By integrating

Mininet with OpenDaylight, professionals can simulate a complete SDN environment.[1] This integration allows for the testing of dynamic network policies, real-time network adjustments, and the overall impact of SDN on network performance and reliability. It provides a sandbox where complex scenarios can be safely executed and analyzed, leading to more robust and reliable SDN solutions.

Additionally, Mininet supports various network applications and tools, extending its usability beyond SDN. It can emulate traditional network setups, IoT environments, and even multi-cloud networks, making it a versatile tool for a broad range of networking projects. Its support for OpenFlow, a widely adopted SDN protocol, further enhances its applicability in modern network research and development.[6] [5]

Mininet's open-source nature and active community support are also significant advantages. The collaborative development environment ensures that Mininet stays updated with the latest networking trends and technologies. Extensive documentation and community forums provide ample resources for troubleshooting and optimization, making it accessible even to those new to network emulation.

Mininet is effective for network emulation but has limitations, especially with MiniEdit, its graphical interface. While MiniEdit simplifies topology creation, it lacks the depth, flexibility, and advanced features of Mininet's command-line interface (CLI), posing restrictions in usability, customization, and performance.[7] [3]

It stands out as a powerful and flexible tool in the network professional's toolkit. Its ability to create realistic network environments, support for SDN controllers like OpenDaylight, and broad applicability make it indispensable for advancing network research, development, and deployment.

IV. MiniEdit

MiniEdit is a graphical user interface (GUI) for Mininet that simplifies the process of creating and managing network topologies.[8] Designed with ease of use in mind, MiniEdit provides a drag-and-drop interface that allows users to construct complex network layouts visually. This intuitive interface makes it accessible to users who may not be familiar with Mininet's command-line interface, lowering the barrier to entry for network experimentation and education.

With MiniEdit, users can effortlessly add hosts, switches, links, and controllers to their network topology by simply dragging and connecting elements on a canvas. This visual approach not only speeds up the topology creation process but also makes it easier to understand and modify existing network configurations. Once the desired topology is designed, MiniEdit can export the configuration to Mininet, enabling immediate emulation and testing.

MiniEdit also supports the customization of network parameters, such as link bandwidth and delay, directly within the GUI. This feature allows for more precise control over the network environment, making it ideal for simulating specific network conditions and conducting detailed performance evaluations. Additionally, MiniEdit's ability to save and load network configurations streamlines repeated testing and iterative development processes.

- **Limited Functionality:** While MiniEdit simplifies the creation and configuration of network topologies, it lacks some of the advanced features and functionalities available in Mininet's command-line interface (CLI). Users may find themselves restricted in terms of what they can configure or customize through the GUI alone.
- **Steep Learning Curve:** Despite its user-friendly interface, MiniEdit still requires users to have a basic understanding of networking concepts and Mininet's operation. New users may need to invest time in learning how to effectively use the tool, especially if they are unfamiliar with network emulation and virtualization concepts.
- **Resource Intensive:** MiniEdit can be resource-intensive, particularly when dealing with large or complex network topologies. Users may experience performance issues or slowdowns, especially on systems with limited computational resources.
- **Limited Support for Advanced Features:** MiniEdit may lack support for some of the advanced features and functionalities available in Mininet, such as custom network protocols, dynamic routing algorithms, or advanced traffic engineering mechanisms. Users requiring these advanced capabilities may need to resort to the Mininet CLI or custom scripts.
- **Dependency on Graphics:** MiniEdit relies on the Graphviz library for topology visualization. Users need to have Graphviz installed on their system to fully utilize MiniEdit's visualization capabilities. Dependency on external libraries can introduce compatibility issues or additional setup requirements.

V. Mininet Remastered

It is needed for a Minnet and a new GUI, Developing a new graphical user interface (GUI) for the Mininet emulator might be necessary or beneficial for several reasons:

- **Enhanced Usability:** The existing GUI tools for Mininet may lack certain features or functionalities that users require for their specific use cases. Developing a new GUI allows for customization and optimization of the user interface to better suit the needs of users, potentially leading to improved usability and user satisfaction.

- **Advanced Functionality:** A new GUI can incorporate advanced features and functionalities that are not available in existing GUI tools. This could include support for custom network protocols, dynamic routing algorithms, traffic engineering mechanisms, or integration with other network simulation or emulation tools.
- **Platform Compatibility:** Existing GUI tools for Mininet may be limited in terms of platform compatibility, running only on specific operating systems or environments. Developing a new GUI with cross-platform support allows users to access Mininet's capabilities from a wider range of devices and operating systems.
- **Performance Optimization:** The performance of existing GUI tools may be suboptimal, particularly when dealing with large or complex network topologies. Developing a new GUI allows for optimization of performance, reducing resource consumption and improving responsiveness, especially on systems with limited computational resources.
- **Integration with Other Tools:** new GUI can be designed to integrate seamlessly with other network simulation or emulation tools, as well as with external monitoring, visualization, or management systems. This integration enhances the overall utility and interoperability of the toolset for network engineers and researchers.
- **Educational Purposes:** Developing a new GUI for Mininet can serve as an educational project for students or researchers interested in network simulation, emulation, or software development. It provides an opportunity to learn about network concepts, software design principles, and graphical user interface development in a practical context.

Analysis of the new GUI "Mininet Remastered":

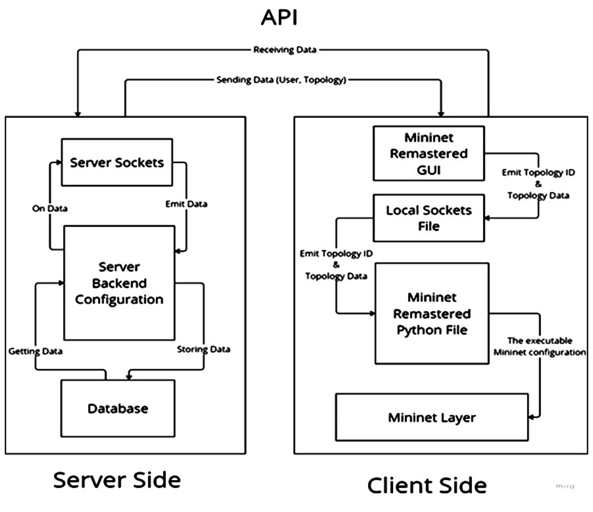
API
Receiving Data
Sending Data (User, Topology)
Server Sockets
On Data
Emit Data
Server Backend Configuration
Getting Data
Storing Data
Database
Mininet Remastered GUI
Emit Topology ID & Topology Data
Local Sockets File
Emit Topology ID & Topology Data
Mininet Remastered Python File
The executable Mininet configuration
Mininet Layer
Server Side
Client Side

Fig. 2: Workflow of proposed GUI

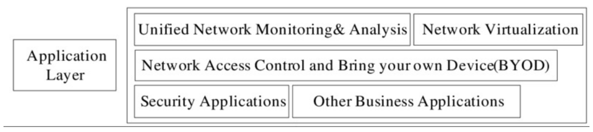
Application Layer
Unified Network Monitoring& Analysis
Network Virtualization
Network Access Control and Bring your own Device(BYOD)
Security Applications
Other Business Applications

Fig. 3: Application Layer Components

Mininet Remastered workflow diagram ensures a seamless interaction between the user and the Mininet environment through a structured GUI that deals with the application layer components, facilitating the configuration, execution, and management of network topologies in a user-friendly manner. The server side handles data management and processing, while the client side provides the interface and executes the Mininet commands, all synchronized through an API. The Mininet GUI application workflow consists of two main components: the Server Side and the Client Side, both of which interact through an API.

1) Server Side:

- **Server Sockets**
  - **Function:** The server sockets are responsible for receiving and emitting data. They act as the communication interface between the client and the server backend.
  - **Interactions:**
    - ∗ On receiving data from the client, the server sockets pass this data to the server backend configuration.
    - ∗ When the server backend has data to send, it is emitted through the server sockets to the client.
- **server Backend Configuration**
  - **Function:** This module processes incoming data and prepares it for storage or further handling. It also retrieves data from the database when required.
  - **Interactions:**
    - ∗ Receives data from the server sockets, processes it, and stores necessary information in the database.
    - ∗ Retrieves data from the database when needed and sends it back to the server sockets for emission.
- **Database**
  - **Function:** The database is the storage layer where all the necessary data (such as user information and topology configurations) is kept.
  - **Interactions:**
    - ∗ Stores data received from the server backend configuration.
    - ∗ Provides data to the server backend configuration when a request is made.

2) Client Side:

- **Mininet Remastered GUI**
  - **Function:** The GUI provides the user interface for interacting with the Mininet application. Users can configure network topologies and interact with the Mininet environment through this interface.
  - **Interactions:**
    - Emits topology IDs and configuration data based on user inputs.
    - Sends this data to the local sockets file for further processing.
- **Local Sockets File**
  - **Function:** Acts as a local intermediary between the GUI and the Mininet remastered Python file
  - **Interactions:**
    - Receives topology IDs and configuration data from the Mininet Remastered GUI.
    - Passes this data to the Mininet remastered Python file.
    - Can also receive data back from the Python file to update the GUI if needed.
- **Mininet Remastered Python File**
  - **Function:** This is the core execution layer for the Mininet network emulation. It interprets the configuration data and executes the necessary commands to set up the network topology.
  - **Interactions:**
    - Receives executable Mininet configurations from the local sockets file.
    - Interacts directly with the Mininet layer to create and manage network topologies.
- **Mininet Layer**
  - **Function:** The actual Mininet environment where network topologies are emulated.
  - **Interactions:** Takes configuration commands from the Mininet remastered Python file to set up and manage the network topology.
- **Interaction through API**
  The API facilitates communication between the server and client sides:
  - **Data Flow:**
    - The server side sends user information and topology configurations to the client side.
    - The client side sends back topology data and configuration details to the server side.
  - **Types of Data:**
    - User information, topology configurations, topology IDs, and related data are exchanged through the API to maintain synchronization between the client and server.

Advantages of the new GUI "Mininet Remastered":

Developing a new graphical user interface (GUI) for the Mininet emulator offers several advantages, including:

- **Improved Usability:** Mininet Remastered provides a more intuitive and user-friendly interface compared to the command-line interface (CLI) of Mininet. This can lower the barrier to entry for users who may not be familiar with networking concepts or command-line tools, making it easier for them to create, configure, and manage network topologies.
- **Enhanced Visualization:** Mininet Remastered offers advanced visualization capabilities, allowing users to create and interact with network topologies in a more visual and intuitive manner. This can help users better understand the structure and behavior of their networks identify potential issues or bottlenecks, and make informed decisions about network design and configuration.
- **Streamlined Workflow:** Mininet Remastered can streamline common tasks and workflows, reducing the time and effort required to perform network emulation and testing. Users can quickly deploy, modify, and tear down network topologies using intuitive controls and visual feedback, improving overall productivity and efficiency.
- **Customization and Extensibility:** Mininet Remastered is designed to support customization and extensibility, it is an open-source code, which allowing users to tailor the interface to their specific needs and preferences. This can include customizing the layout and appearance of the GUI, adding new features or functionalities, or integrating with other tools and technologies to extend its capabilities.
- **Cross-Platform Compatibility:** Mininet Remastered offers cross-platform compatibility, allowing users to access Mininet's capabilities from a wide range of devices and operating systems. This can increase accessibility and flexibility, enabling users to work with Mininet on their preferred platform without compatibility issues or limitations.
- **Integration with External Tools:** Mininet Remastered can be integrated with external tools and technologies, such as network monitoring and management systems, or cloud services. This integration can enhance the overall utility and interoperability of the GUI, providing users with a comprehensive solution for network development, testing, and management.
- **Educational Tool:** Mininet Remastered can serve as an educational tool for teaching networking concepts and principles. Its intuitive interface and visual feedback make it suitable for use in classroom settings or for self-study, helping students learn about network emulation, virtual-

ization, and software-defined networking (SDN) in a practical and hands-on manner.

Risk Management:
Developing a new graphical user interface (GUI) for the Mininet emulator entails various risks, ranging from technical challenges to resource constraints. Here are some potential risks and corresponding risk management strategies:

- **Technical Complexity:**
  - **Risk:** GUI development for a complex system like Mininet may involve intricate technical requirements, including network topology visualization, configuration management, and integration with the Mininet core.
  - **Risk Management Plan:** Conduct a thorough analysis of the technical requirements before starting development. Break down the project into manageable tasks and allocate sufficient time for research and prototyping. Consider leveraging existing GUI frameworks or libraries to streamline development.
- **Compatibility Issues:**
  - **Risk:** The GUI may encounter compatibility issues with different operating systems, browsers, or Mininet versions, leading to inconsistent performance or functionality.
  - **Risk Management Plan:** Test the GUI across multiple platforms and Mininet configurations to identify compatibility issues early in the development process. Establish clear compatibility requirements and document supported environments. Provide user guidance for resolving compatibility issues and offer regular updates or patches as needed.
- **Resource Constraints:**
  - **Risk:** Limited resources, including time, budget, and expertise, may pose challenges during GUI development, resulting in delays, cost overruns, or compromised quality.
  - **Risk Management Plan:** Define realistic project timelines, budgets, and resource allocations based on a thorough assessment of project requirements and constraints. Prioritize essential features and functionalities to ensure the efficient use of resources. Consider outsourcing certain tasks or seeking additional expertise if necessary.
- **User Adoption and Feedback:**
  - **Risk:** The new GUI may fail to meet user expectations or address their needs effectively, leading to low user adoption rates or negative feedback.
  - **Risk Management Plan:** Involve end-users, such as network engineers, researchers, or educators, in the development process through user interviews, surveys, or usability testing. Gather feedback iteratively and incorporate user suggestions and preferences into the GUI design. Provide comprehensive documentation, tutorials, and user support to facilitate user onboarding and engagement.
- **Security Vulnerabilities:**
  - **Risk:** The GUI may introduce security vulnerabilities, such as input validation flaws, authentication bypasses, or data leakage, compromising the integrity and confidentiality of Mininet environments.
  - **Risk Management Plan:** Implement robust security measures, including input validation, authentication mechanisms, encryption, and access controls, to mitigate potential security risks. Conduct regular security audits and penetration testing to identify and address vulnerabilities proactively. Stay informed about security best practices and industry standards to maintain a secure development environment.
- **Lack of Community Support:**
  - **Risk:** The new GUI may struggle to gain traction within the Mininet community, resulting in limited adoption and support from users and contributors.
  - **Risk Management Plan:** Foster community engagement and collaboration by actively promoting the new GUI through social media, forums, conferences, and other channels. Solicit feedback from community members and incorporate their suggestions into the development roadmap. Contribute to open-source projects, share code samples, and provide documentation to encourage community participation and contribution.

VI. Conclusion

The development of a network topology application with a graphical user interface (GUI) for the Mininet emulator represents a significant advancement in network management and administration. This tool aimed to simplify the complex task of creating and managing network topologies, thereby making it more accessible and user-friendly for network engineers and administrators.

Throughout the paper, we focused on energizing the robust capabilities of the Mininet emulator. By integrating a GUI, we have transformed the traditionally command-line-driven process into an intuitive visual experience. This interface allows users to easily visualize, design, and manage network topologies through simple drag-and-drop functionalities and real-time feedback, reducing the learning curve and minimizing the potential for errors.

Mininet Remastered represents a significant step forward in network emulation by integrating a user-friendly graphical interface with the core functionalities of the popular Mininet platform. This advancement addresses one of the primary challenges faced by network researchers and educators—the complexity and steep learning curve associated with command-line interfaces. By offering a more intuitive GUI, Mininet Remastered lowers the barrier to entry, making it accessible to a wider audience, from students and instructors in academic settings to professionals seeking a practical tool for network prototyping and testing.

The development of Mininet Remastered opens new possibilities for interactive and visualized network management and experimentation, facilitating more effective learning and exploration of networking concepts. Early feedback suggests that users find the GUI enhances their productivity and improves their understanding of network topologies and behaviors. Moving forward, we aim to expand Mininet Remastered's capabilities by incorporating advanced visualization options, support for more complex protocols, and features that enable real-time data analysis and sharing.

In summary, Mininet Remastered provides an approachable and powerful tool for network emulation that aligns with the needs of modern users, emphasizing usability without sacrificing functionality. We believe that this enhanced accessibility will not only broaden the adoption of network emulation tools but also inspire further innovations in educational and professional network environments. This project not only addresses current needs but also sets a foundation for future enhancements in network emulation solutions.

VII. Future Work

While Mininet Remastered has made network emulation more accessible through a user-friendly GUI, there are several areas where the tool can be further improved and expanded. Future work on Mininet Remastered could focus on the following aspects:

- **Advanced Network Visualization:** Adding more sophisticated visualization tools, such as real-time packet flow monitoring and traffic heat maps, would provide users with a deeper understanding of network behavior. Interactive visualizations could help users analyze complex network topologies and identify bottlenecks or points of failure more effectively.
- **Integration with Cloud and SDN Platforms:** To increase its versatility, Mininet Remastered could be integrated with popular cloud services and Software Defined Networking (SDN) controllers. This would enable users to simulate, test, and deploy network configurations that mirror real-world cloud and SDN environments, making it a valuable tool for researchers and professionals working on hybrid network models.
- **Support for More Complex Protocols and IoT Networks:** Expanding support for protocols beyond the basic TCP/IP stack would allow users to model more complex and specialized networks, including IoT and mobile networks. Incorporating lightweight, resource-constrained IoT devices and protocols would enable Mininet Remastered to serve as a comprehensive tool for simulating next-generation networks.
- **Enhanced Collaboration Features:** Adding features that allow users to save, share, and collaborate on network topologies and simulations would benefit educational and team-based research settings. Version control integration and multi-user editing capabilities could further streamline the collaborative process for network design and analysis.
- **Performance Optimization and Scalability:** While Mininet is efficient for small to medium-scale network simulations, optimizing Mininet Remastered to handle larger topologies with minimal performance impact would broaden its applicability. This may involve leveraging distributed computing resources or GPU acceleration to manage high-volume simulations.
- **Machine Learning Integration for Automated Network Analysis:** Leveraging machine learning algorithms to analyze and predict network performance under different configurations could add a powerful layer of intelligence to Mininet Remastered. For instance, ML models could suggest optimal network configurations or help detect anomalous behaviors automatically.

In summary, future developments of Mininet Remastered can transform it into a more powerful, flexible, and collaborative tool. By incorporating these advanced features, Mininet Remastered could become an indispensable resource for network research, education, and prototyping, further bridging the gap between conceptual learning and practical application.

References

[1] Sreebha Bhaskaran and M.Supriya. "Study on Networking Models of SDN using Mininet and MiniEdit". In: *2024 2nd International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT)* (2024).

[2] Rogério Leão Santos De Oliveira et al. "Using mininet for emulation and prototyping software-defined networks". In: *2014 IEEE Colombian conference on communications and computing (COLCOM)* (2014).

[3] Stefan Geiß et al. "Mininet-based SDN Testbed for Research and Development". In: *University of Rostock* (2015).

[4] Akram Hakiri et al. "Software-defined Networking: Challenges and Research Opportunities for Future Internet". In: *Computer Networks* 75 (2014), pp. 453–471.

[5] Japinder Singh Kaur Karamjeet and Navtej Singh Ghumman. "Mininet as software defined networking testing platform". In: *International conference on communication, computing & systems (ICCCS)* (2014).

[6] Japinder Singh Kaur Sukhveer and Navtej Singh Ghumman. "Network programmability using POX controller". In: *ICCCS International conference on communication, computing & systems* 138 (2014).

[7] Giuseppe Di Lena et al. "Distrinet: A Mininet Implementation for the Cloud". In: *HAL Inria* (2020).

[8] Oleksandr Romanov et al. "SDN network modeling using the GUI MiniEdit". In: *2022 IEEE 16th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET)* (2022).