Integrating Reality: A Hybrid SDN Testbed for Enhanced Realism in Edge Computing Simulations

Amar Almaini
Department of computer science
Deggendorf Institute of Technology
Deggendorf, Germany

Tobias Koßmann
Department of computer science
Deggendorf Institute of Technology
Deggendorf, Germany

Jakob Folz
Department of computer science
Deggendorf Institute of Technology
Deggendorf, Germany

Martin Schramm
Department of computer science
Deggendorf Institute of Technology
Deggendorf, Germany

Michael Heigl
Department of computer science
Deggendorf Institute of Technology
Deggendorf, Germany

Ahmed Al-Dubai
School of Computing
Edinburgh Napier University
Edinburgh, UK

***Abstract*—Recent advancements in Software-Defined Networking (SDN) have facilitated its deployment across diverse network types, including edge networks. Given the broad applicability of SDN and the complexity of large-scale environments, establishing a comprehensive real-world test environment is both challenging and expensive. To circumvent these obstacles, software-based simulations are typically employed to validate solutions prior to real-world deployment. However, these simulations often do not incorporate real-time hardware data, limiting their realism. This paper introduces a novel hybrid SDN simulation testbed that integrates real hardware data within a Mininet-emulated network, addressing this limitation. To demonstrate the efficacy of our hybrid testbed, we present a specific scenario involving the dynamic allocation of edge resources to various client requests through a machine learning approach. This scenario focuses on detecting LiDAR spoofing attacks within automotive systems. Additionally, our hybrid testbed facilitates the generation and replication of new datasets for tailored scenarios, enhancing research capabilities in more intricate contexts.**

***Index Terms*—Hybrid SDN Simulation, Mininet, Network Testbed Design, Software-Defined Networking , Edge Computing**

I. Introduction

The integration of SDN into diverse environments, such as edge networks [1], necessitates the creation of robust testing environments. These environments validate SDN-based system proposals before deployment in real-world scenarios. However, the challenge lies in emulating large-scale networks, which are both complex and costly to replicate physically for testing purposes. Consequently, the use of dedicated software simulators has become a preferred testing approach [2]. Within the SDN framework, it is crucial for a simulator to support scenario implementation across all SDN architectural layers: the data plane, control plane, and application plane. The data plane involves devices like desktop computers or edge devices that forward network traffic and connect to switches. The control plane manages these devices, setting the operational rules, while the applications plane encompasses end-user applications that utilize the data from the underlying layers. This paper focuses on designing a simulation testbed tailored to the diverse applicability domains of SDN architecture. Our proposed testbed is a hybrid environment that integrates software-generated and real-time traffic captured using actual hardware components. This approach not only enhances the realism of the simulations but also supports the analysis of traffic patterns and volumes in a controlled setting, which is critical for preparing network deployments in production environments. Mininet, a widely used network emulator in SDN research [3], forms the basis of our simulation implementations. The hybrid nature of the testbed, combining Mininet-based solutions with real hardware data, is crucial for replicating large-scale network behaviors, thereby providing significant value to network research [4].

The key contributions of this research are detailed below:

- Development of a hybrid simulation testbed that merges simulated and real-time data to produce realistic network simulations.
- Addressing the varied nature of SDN deployment environments through a testbed design that emphasizes scalability, parallel computation, and portability, enhancing the quality and relevance of experimental outcomes.
- Technical elaboration and practical application of hybrid SDN simulations in a real-world scenario, specifically the dynamic allocation of edge resources using a machine learning approach to detect LiDAR spoofing attacks in automotive systems.

This paper also introduces new possibilities for generating and replicating specific data sets from real-time sensor data, thereby expanding the available research resources and contributing to the scientific community's collective knowledge.

The structure of this paper is organized as follows: Section 2 presents a review of the literature relevant to our work. Section 3 describes the tools employed in our Testbed, specif-

ically focusing on Resource Allocation and Shadow Analyzer. Section 4 details the various components of our Testbed setup, including Hardware Resources, the Virtualized Environment, and the Client-Controller-Resource Workflow. Section 5 and subsequent subsections provide a baseline analysis conducted locally on a single device without networking, assess the performance of our proposed Testbed, particularly in terms of communication delays, and further evaluate our edge nodes. The paper concludes with a summary and future research directions in Section 6.

II. Related Work

Mininet is a useful tool for emulating various parts of a network, such as hosts and switches. It offers a simple solution for creating, modifying, and rolling out new topologies via script, which enables rapid development at low cost. Additionally, it provides built-in commands that allow users to test network functionality, including standard options like the ability to ping hosts and generate network traffic using tools like iperf to simulate network floods or other traffic patterns. However, it has limitations concerning easy scalability and simultaneous deployment on multiple machines [5], [6]. ComNetsEmu offers a container-based extension to the existing Mininet. It removes the CPU limitations of traditional Mininet hosts by employing fully customizable containers, resulting in a nested virtualization architecture. The introduction of these containers enables users to run more specialized tasks on hosts. Since ComNetsEmu is built upon Mininet, it inherits the same limitations, such as being restricted to emulating networks on a single host [7]. To address this limitation, Wette et al. [8] expanded the existing Mininet solution to Maxinet, with the goal of emulating large datacenter networks with several thousand nodes. This was achieved by running the emulation on multiple physical machines connected through GRE tunnels [9]. By leveraging the advantages of Maxinet, it was possible to emulate large-scale datacenter behavior with over 3600 individual nodes using just 12 physical machines. Distrinet also employs a distributed approach by implementing Mininet over multiple hosts while maintaining full compatibility with the known Mininet API. This distribution across multiple physical machines is achieved through automatic cloud provisioning and tunneling over VXLAN [10]. Furthermore, it offers the possibility of vLink limitations to simulate more realistic bandwidth and delay [11]. To enable a better simulation of real hardware and not just the replay of previously recorded logs, Buzura et al. [12] proposed a hybrid solution using real sensor data in the emulation of a Mininet network to provide a more realistic simulation environment. This solution does not incorporate real network hardware into a Mininet network. Tools like Mininet or Maxinet only focus on an emulated approach and do not offer a hybrid solution to incorporate real hardware as a host into a simulation/testbed. Windisch et al. [13] proposed a hybrid solution by letting a master coordinator generate an intermediate representation layer. Worker coordinators can then autonomously convert this layer into deployment scripts, albeit with a large overhead, to enable the deployment of different topologies.

III. Resource Allocation and Shadow Analyzer: Tools for Demonstrating the Testbed Functionality

In the context of Edge Computing (EC), our previous research introduced innovative technologies to address the complexities of resource management and security in dynamic environments. In this paper, these technologies are utilized to demonstrate the functionality of our newly developed testbed, which provides a real hardware environment for testing and validation. First, we proposed a reinforcement learning-based framework for resource allocation, which dynamically assigns resources to tasks, optimizing service delays and balancing resource utilization. Unlike conventional methods that rely on static data and centralized control, this framework utilizes a hierarchical organization of AI controllers that learn and adapt through interaction with the environment. This approach allows for a more flexible and responsive system, capable of adjusting to changes such as device mobility and varying task requirements. Next, we developed the Shadow Analyzer, a technology critical for enhancing the safety and efficiency of modern transportation systems via EC. This tool plays a pivotal role in Vehicle-to-Thing communication, particularly in managing emergency road scenarios and improving traffic safety through advanced vehicle detection and tracking methods. By processing data at the edge, the Shadow Analyzer minimizes latency and enhances real-time responses. Furthermore, it incorporates neural network-based object shadow verification to tackle security threats such as LiDAR spoofing attacks, ensuring the reliability and integrity of autonomous vehicles. Both technologies, developed previously, are utilized in this paper to showcase the functionality of our testbed, which was specifically designed to test these solutions in a real hardware environment. This testbed provides a practical platform for evaluating and demonstrating the performance and effectiveness of our EC solutions in addressing key challenges in resource management and vehicular safety.

IV. Testbed Setup for Resource Allocation Validation

In this study, we illustrate the capabilities of our experimental testbed through a resource management scenario, leveraging a hybrid configuration. Our testbed integrates a Mininet environment to emulate client behaviors, a virtualized SDN controller based on the Ryu platform, and various hardware resources interconnected within the network (see Figure 1).

A. Hardware Resources

Our primary objective was to perform a variety of computational tasks ranging from basic arithmetic to complex automotive applications, such as the detection of Lidar spoofing attacks. To provide a realistic representation, we employed two distinct computational platforms with unique capabilities. The first platform was the Raspberry Pi 3 B+, a Single Board

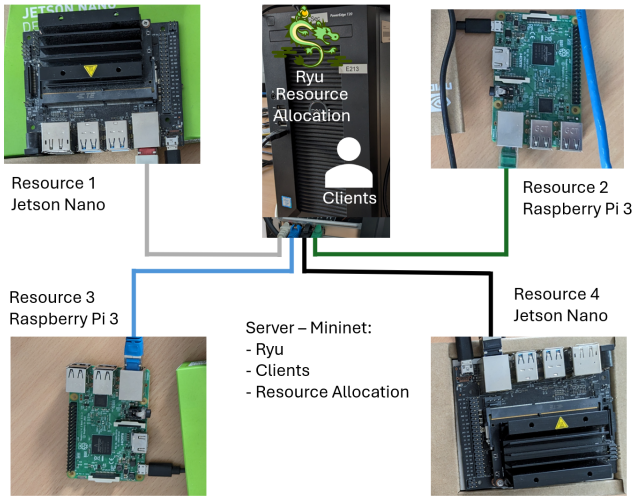

Fig. 1. Testbed components

Computer (SBC) featuring a Broadcom BCM2837B0, Cortex-A53 64-bit SoC @ 1.4 GHz, and 1 GB of RAM. Noteworthy for its compact size and energy efficiency, this platform exemplifies less powerful computational resources. Conversely, the Nvidia Jetson Nano represented a more robust computational resource. It is equipped with an ARM® Cortex®-A57 processor, a 128-core Maxwell GPU that supports CUDA, and 4 GB of RAM. Notably, it maintains a size and power consumption similar to that of the Raspberry Pi 3 B+. CUDA, an API by NVIDIA, facilitates the offloading of computational tasks to the GPU, allowing for significantly faster processing compared to traditional CPU computations. To illustrate these differences, we analyzed comparative performance data. The Shadow Analyzer software, when operated solely on the Raspberry Pi's CPU, required approximately 390 ms to complete a specific task (as shown in Figure 2). In stark contrast, the Jetson Nano, utilizing its CUDA-enabled GPU, completed the same task in just 190 ms. This demonstrates the substantial performance enhancement achievable with advanced GPU computing.

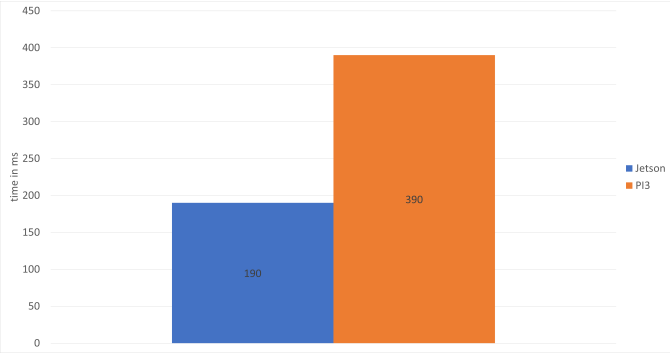

Fig. 2. GPU vs CPU Performance.

B. Virtualized Environments

In the virtualized environment described, we achieve a comprehensive mapping of the network using Mininet. This mapping includes network nodes/switches, distinct network pathways, and clients that exclusively request hardware resources, as well as interfaces that bridge real and virtualized hardware. Dedicated physical network ports are directly connected to specific virtualized switches, forming a bridge between the simulated network and the actual hardware. The network is orchestrated by an SDN controller that utilizes Ryu [14]. Our proposed resource allocation strategy has been integrated into this controller. Figure 3, visualized using the Mininet Topology Visualizer, depicts a representative setup with two clients and four resources. The network employs a straightforward tree topology; the first layer features a central main switch (msw0), while the second layer splits into two switches. One switch connects to resources (rsw0), and the other to the virtualized clients (csw0). Although there is potential for connections to virtualized resources, this hybrid setup primarily focuses on hardware resources. The depicted resources, each connected via a four-port network card, with each port mapped to a specific port on rsw0, are tangible. In this scenario, Jetson Nanos are utilized for rs1 and rs3, while Raspberry Pi 3s are used for rs2 and rs4. The SDN Controller, based on Ryu and not shown in the figure, is an OpenFlow Controller that manages the entire network, including switches, network routes, and other components.

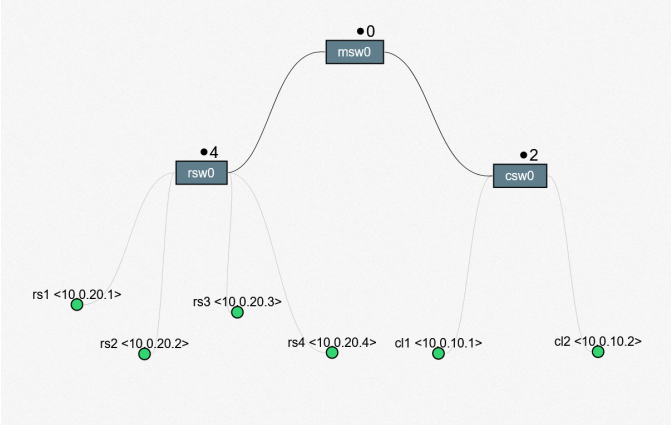

Fig. 3. Visual Representation of a Hybrid Virtual-Physical Network with SDN Control and Direct Hardware Resource Mapping.

C. SDN-Based Resource Allocation and Client Request Flow

Figure 4 illustrates the flow of a client request and its corresponding response. Initially, the client issues a request, which is then directed to the resource allocation module operating on the SDN controller. Within the controller, a predictive analysis is conducted to determine the most suitable resource to address the request. Subsequently, the request is relayed to the identified resource for processing. The processed response is then channeled back to the resource allocation module,

which subsequently forwards it to the originating client. A notable advantage of this indirect communication method, as compared to direct communication, is that the client remains unaware of specific resource parameters such as suitability, utilization, and IP address. All requisite information resides centrally within the resource allocation module. Therefore, it is sufficient for the client to direct its inquiry to the SDN controller.

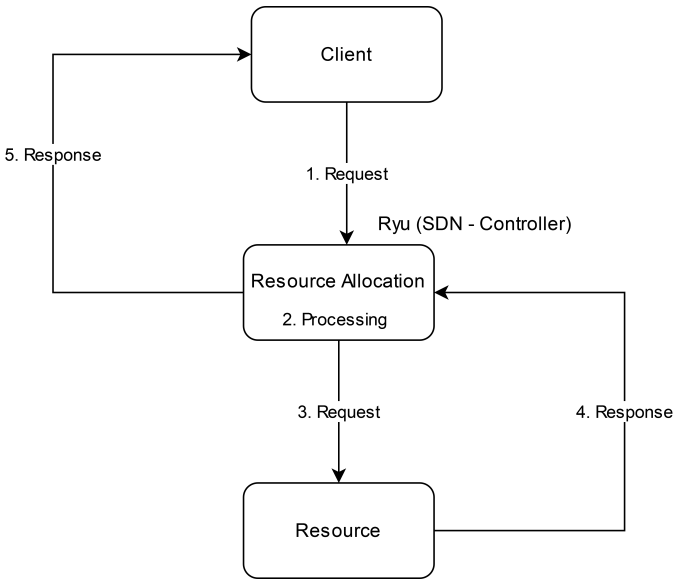

Fig. 4. Client-Controller-Resource Workflow.

V. Evaluation

To evaluate the efficacy of our testbed, we examined various configurations.

A. Performance Benchmarking Using a Single Local Device

The primary configuration involved using a single local device. This scenario represents cases where the prediction of LiDAR spoofing attacks is executed directly on the dedicated hardware of the vehicle. This configuration serves as a benchmark for comparison with other measurements. To emulate this scenario, a Jetson device was employed locally as the dedicated hardware for the task. Table I presents the average prediction time for a single image. We conducted ten test runs, each encompassing 1,000 images. The average prediction time for each run was calculated, with the overall mean across all runs presented in the table as the reference value.

B. Remote Device Evaluation at the Edge and its Communication Delays

In this phase of our evaluation, we adopted a methodology similar to that described previously, with a key modification concerning the location of the device. Unlike the initial experiment where the device was local and integrated into the vehicle, in this phase, the device is positioned remotely at the edge. This configuration involved direct client interactions with a single remote resource, bypassing the resource allocation framework. This approach could lead to increased communication delays. The data in Table II show that the average response time under these conditions is slightly higher compared to the local setup. It is important to note that these experiments were conducted in a controlled laboratory environment, which may not fully represent the communication latency experienced in real-world applications.

TABLE I
Average Prediction Time for LiDAR Spoofing Attack Detection on Jetson Device

Round	Time (ms)
1	191,792
2	190,510
3	190,554
4	191,215
5	190,108
6	190,277
7	190,050
8	189,924
9	189,786
10	190,611
Average	**190,483**

TABLE II
Average Prediction Time for LiDAR Spoofing Attack Detection on Remote Jetson Device

Round	Average Time per Image in ms
1	190,277
2	190,050
3	189,924
4	189,786
5	191,190
6	190,611
7	190,678
8	191,297
9	191,067
10	192,141
Average	**190,702**

C. Resource Allocation Between Jetson and Raspberry Pi

In this section, we investigate the role of mediation in the resource allocation framework by utilizing two distinct edge-located resources: a Jetson and a Raspberry Pi. These devices were chosen to represent the varied computational capacities available at the edge. Table III presents the results from ten iterations, with each iteration processing 1,000 images, in alignment with the testing methodology discussed in previous sections. Columns two and three of Table III

display the distribution of images between the two devices in each iteration. The data shows that the Jetson, being the more powerful device, is preferentially selected by the framework over the Raspberry Pi. On average, about 70% of all images are processed by the Jetson, as depicted in Figure 5. This strategy of allocation significantly enhances the overall processing efficiency. As indicated in the fourth column of Table III, the average processing time per image improved by 11.79% relative to the baseline established in the initial test. The final column provides details on the average time taken by the resource allocation framework in each iteration to determine the optimal resource for the current task. The minimal time recorded indicates that its impact on the total processing duration is negligible.

TABLE III
DISTRIBUTION OF IMAGES AND PROCESSING TIME ACROSS 2 EDGE DEVICES

Round	Jetson 1	Raspberry Pi	Time/Image (ms)	Pred. Time (ms)
1	709	291	160,823	8,949
2	688	312	183,993	8,825
3	697	303	174,706	8,413
4	676	324	161,024	8,475
5	689	311	160,879	9,223
6	709	291	160,831	8,943
7	723	277	160,804	8,874
8	804	196	161,123	9,398
9	701	299	167,923	8,845
10	793	207	188,211	9,101
Average	**718.9**	**281.1**	**168,032**	**8,905**

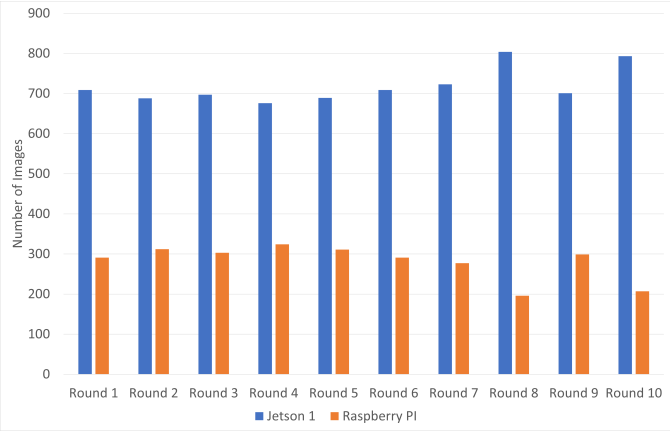

Fig. 5. Distribution of Images Between Jetson and Raspberry Pi.

D. Image Distribution and Processing Time with Four Edge-located Resources

In this section, we extend our examination from the prior experiment by including four edge-based computing devices: two NVIDIA Jetsons and two Raspberry Pis. Our goal is to evaluate how the increase in available resources influences the distribution of image processing tasks among these devices and the average image processing time, ultimately impacting the overall processing duration. The results from ten iterations, each processing 1,000 images, are presented in Table IV. This testing approach is consistent with methodologies applied in previous sections. Columns two through five of Table IV illustrate how images are allocated across the four devices in each iteration. Notably, the data indicates a distinct preference within the framework for allocating approximately 62% of all images to the Jetsons. This allocation pattern is further depicted in Figure 6. The sixth column of Table IV shows a 49.94% improvement in the average processing time per image compared to the baseline established in the initial experiment, and it is 43.25% more efficient than the prior experiment that utilized only two resources. The last column provides insights into the average time required by the resource allocation framework to determine the optimal resource for processing each image in each iteration.

TABLE IV
DISTRIBUTION OF IMAGES AND PROCESSING TIME ACROSS 4 EDGE DEVICES

Round	Jet. 1	RPI 1	Jet. 2	RPI 2	Avg. Time (ms)	Pred. Time (ms)
1	290	158	364	188	100,450	14,638
2	272	175	341	212	93,119	14,710
3	263	165	369	203	91,599	14,617
4	279	143	367	211	93,820	14,822
5	239	187	366	208	93,706	16,071
6	269	168	356	207	93,131	14,809
7	266	181	323	230	100,129	14,403
8	284	186	341	189	98,939	14,695
9	267	163	357	213	94,179	14,893
10	288	151	350	211	94,531	14,734
Average	**271.7**	**167.7**	**353.4**	**207.2**	**95,360**	**14,839**

VI. CONCLUSION AND FUTURE WORK

This paper has presented a comprehensive exploration of our hybrid SDN simulation testbed, which successfully integrates real-time hardware into a Mininet-emulated network. This integration is crucial for adding realism and depth to network simulations, which are essential in the face of the increasing complexity and broad applicability of SDN across various network types, especially in edge computing scenarios. Our testbed not only meets the challenge of creating a realistic emulation environment that can facilitate the study of SDN but also expands the capabilities for real-world applications.

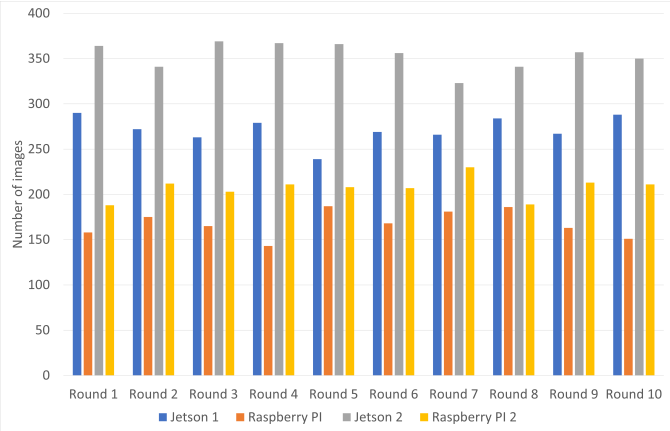

Fig. 6. Distribution of Images Between Jetsons and Raspberry Pis.

Through extensive testing using scenarios that involve dynamic resource allocation to counter LiDAR spoofing attacks in automotive systems, our testbed has proven effective. It demonstrates significant adaptability in managing diverse computing resources at the edge, an essential feature given the critical nature of the automotive applications tested. The resource allocation framework integrated within the SDN controller has shown a high level of efficiency in utilizing the available computational resources. This efficiency is highlighted by the system's ability to optimize task completion times across different computational platforms, from less powerful devices like Raspberry Pi 3 B+ to more robust systems such as the Nvidia Jetson Nano. Furthermore, the testbed's scalability, parallel computation capabilities, and portability align seamlessly with the design goals initially set for this project. The successful deployment and performance of the testbed validate our approach and provide a solid foundation for future research. This setup not only supports ongoing improvements in network management and attack mitigation strategies but also contributes valuable insights into the operational dynamics of edge computing environments. The implications of this study are vast, offering numerous avenues for future work. For instance, further research could explore the integration of additional types of edge devices or expand the testbed's capabilities to include more complex network scenarios and larger datasets. These enhancements could lead to even more sophisticated solutions to network management and security challenges in various domains beyond automotive systems. In conclusion, the development and validation of this hybrid SDN simulation testbed mark a significant step forward in the field of network research, particularly in the context of edge computing. It stands as a testament to the potential of combining traditional emulation techniques with real-time data to create more accurate, reliable, and flexible network testing environments.

References

[1] A. Liatifis, T. Lagkas, G. P. Katsikas, A. Bujari, V. Argyriou, A. Triantafyllou, A. Lytos, A.-A. A. Boulogeorgos, and P. Sarigiannidis, "Evaluating sdn applicability in the edge," in *2023 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE, 2023, pp. 1571–1575.

[2] H. Yang, J. Ivey, and G. F. Riley, "Scalability comparison of sdn control plane architectures based on simulations," in *2017 IEEE 36th International Performance Computing and Communications Conference (IPCCC)*. IEEE, 2017, pp. 1–8.

[3] R. L. S. De Oliveira, C. M. Schweitzer, A. A. Shinoda, and L. R. Prete, "Using mininet for emulation and prototyping software-defined networks," in *2014 IEEE Colombian conference on communications and computing (COLCOM)*. Ieee, 2014, pp. 1–6.

[4] J. A. García-García, J. G. Enríquez, M. Ruiz, C. Arevalo, and A. Jiménez-Ramírez, "Software process simulation modeling: systematic literature review," *Computer Standards & Interfaces*, vol. 70, p. 103425, 2020.

[5] R. L. S. de Oliveira, C. M. Schweitzer, A. A. Shinoda, and L. R. Prete, "Using mininet for emulation and prototyping software-defined networks," in *2014 IEEE Colombian Conference on Communications and Computing (COLCOM)*, 2014, pp. 1–6.

[6] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, ser. Hotnets-IX. New York, NY, USA: Association for Computing Machinery, 2010. [Online]. Available: https://doi.org/10.1145/1868447.1868466

[7] Z. Xiang, S. Pandi, J. Cabrera, F. Granelli, P. Seeling, and F. H. P. Fitzek, "An open source testbed for virtualized communication networks," *IEEE Communications Magazine*, vol. 59, no. 2, pp. 77–83, 2021.

[8] P. Wette, M. Dräxler, A. Schwabe, F. Wallaschek, M. H. Zahraee, and H. Karl, "Maxinet: Distributed emulation of software-defined networks," in *2014 IFIP Networking Conference*, 2014, pp. 1–9.

[9] G. Dommety, "Key and Sequence Number Extensions to GRE," RFC 2890, Sep. 2000. [Online]. Available: https://www.rfc-editor.org/info/rfc2890

[10] M. Mahalingam, D. Dutt, K. Duda, P. Agarwal, L. Kreeger, T. Sridhar, M. Bursell, and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks," RFC 7348, Aug. 2014. [Online]. Available: https://www.rfc-editor.org/info/rfc7348

[11] G. Di Lena, A. Tomassilli, D. Saucez, F. Giroire, T. Turletti, and C. Lac, "Distrinet: a mininet implementation for the cloud," *SIGCOMM Comput. Commun. Rev.*, vol. 51, no. 1, p. 2–9, mar 2021. [Online]. Available: https://doi.org/10.1145/3457175.3457177

[12] S. Buzura, A. Peculea, B. Iancu, E. Cebuc, V. Dadarlat, and R. Kovacs, "A hybrid software and hardware sdn simulation testbed," *Sensors*, vol. 23, no. 1, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/1/490

[13] F. Windisch, K. Abedi, T. Doan, T. Strufe, and G. T. Nguyen, "Hybrid testbed for security research in software-defined networks," in *2023 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, 2023, pp. 147–152.

[14] R. team, *RYU SDN Framework - English Edition*, ser. Release 1.0. RYU project team, 2014. [Online]. Available: https://books.google.de/books?id=JC3rAgAAQBAJ